Edge Intelligence as a Service

Philipp Raith
Schahram Dustdar
Distributed Systems Group, TU Wien
Vienna, Austria
lastname@dsg.tuwien.ac.at

***Abstract*—Edge Intelligence is the umbrella term for new types of applications, which are being created due to the advent of Internet of Things and the resulting Edge Computing paradigm. Computing resources are pushed to the edge of the network to overcome the massive amounts of generated data, enable ultra low latency applications and guarantee privacy. Edge Intelligence use cases are based on context-aware AI applications and generate new knowledge from heterogeneous data sources. While the promise of these new applications sounds appealing, the reality is that we are still in the infant stage of building such platforms. We motivate the design of this platform by presenting a motivating use case that spans from Food Computing to Smart Health. Based on this, we identify main tasks encountered in the development of AI applications, describe issues related to Edge Intelligence, and present our vision of an Edge Intelligence as a Service platform.**

***Index Terms*—Edge Intelligence, Edge Intelligence as a Service, Elasticity, Edge Computing, Food Computing, Elasticity as a Service**

I. Introduction

We have seen the rise of highly capable AI in recent years, powered by computational capabilities of hardware accelerators, and the consequential research of neural network architectures. AI starts to infiltrate our daily life, covering all important aspects (i.e., health, cities, agriculture). Applications range from personalized food recommendations [1], real-time video analytics [2] and irrigation scheduling predictions [3]. Important characteristics are: context-awareness, ultra-low-latency, large amounts of sensor data, and confidentiality. The emerging Edge Intelligence (EI) paradigm fulfills all the conditions by extending the currently popular cloud-centric infrastructure to the edge of the network. Edge Intelligence is meant to build an unified platform for *AI on edge* applications with yet unseen capabilities by intelligently leveraging resources in near proximity to users [4]. To train models with context in mind, we need to deploy applications where the data originates from. Zhou et al. [5] define six levels of Edge Intelligence characterizing where training and inference is happening. While *Level 1* considers a central training in cloud and using resources at the edge for inference, *Level 6* imagines all tasks to happen on user devices which reduces the amount of data offloading to a minimum and guarantees confidentiality of personal data. In contrast to *AI on edge*, *AI for edge* focuses on developing strategies for platforms to intelligently manage applications as intended by users [4]. Intentions can have different forms and are not limited to guarantee service metrics (i.e., response time) but extend to providing relevant data and context-aware execution. Therefore, Edge Intelligence as a Service platforms (EIaaS) need to allow developers not only to specify service metrics but also to add contextual information, such that intelligent strategies transparently manage deployments.

In general, the platform can be envisioned by deploying a sensing and compute fabric [6]. The former represents the abundance of deployed sensors (i.e., IoT) while the latter acts on the emitted data. The computing fabric is based on the Edge Computing (EC) paradigm, which pushes centralized resources from the cloud to the edge of the network [7]. By pushing resources closer to the users, we can mitigate the issue of the increasing bandwidth needs and can process information and requests immediately. Unfortunately, unleashing the promised potential of Edge Computing is hard and stems from several factors: heterogeneous compute units, offering different capabilities, network distance playing a crucial role to guarantee the stringent latency requirements, yet unknown infrastructure governance, legal policies, context-awareness etc. [8]. To overcome the burden of managing edge infrastructures, accessible platforms are necessary to let people make full use of Edge Intelligence, and therefore allow us to enter the new stage in the cyber-human lifecycle [6], [9]. Transparent deployment guarantees access to AI for businesses without dedicated resources for developing custom solutions and creates an opportunity to generate a wealth of diverse and novel AI applications.

Therefore, the goal of this paper is to envision an end to end platform that pushes the democratization of AI further, by giving non-experts the opportunity to build EI applications. We discuss in detail a motivating use case, centered around a business that produces food and is located in Smart City, Health, and Agriculture as well as Food Computing. The use case showcases elastic properties, data sharing across domain boundaries and helps us outline requirements for an EIaaS platform. Besides showing requirements, it also depicts the possibilities for businesses to leverage AI in a meaningful way to improve quality for themselves but also customers. We identify four main tasks encountered when deploying AI models and highlight for each of them problems that arise in EI and need to be tackled. Based on the previous investigation of use cases, application requirements and tasks, we outline the general architecture and user interaction possibilities. The proposed platform builds on the industry-proven MLOps concept [10], [11], and extends it to enable the lifecycle management of

EI applications. Additionally, we discuss possible implementations of this platform and identify important requirements that the underlying orchestration services must fulfill.

The rest of the paper is structured as follows: in Section II we introduce the necessary background and relevant concepts, accompanied with related work. Section III is dedicated to portraying our use case study by highlighting applications, needed data sources, domains and, elasticity requirements. Upon analysis of possible applications, we describe in Section IV the four main tasks in AI and draw attention to problems that we will inevitably encounter in the future. Based on this, we can envision a plausible platform and afterwards give details to important technical problems encountered in Edge Intelligence in Section V. We outline limitations and future work in Section VI and conclude the work in Section VII.

II. Background & Related Work

A. Edge Intelligence

Edge Intelligence uses data produced at the edge of the network by applying AI applications on it. Edge Computing infrastructures help us solve various issues that arise (i.e., privacy concerns) in EI by pushing resources to the edge of the network. Zhou et al. [5] state in their survey that the term EI is not yet fully defined, but one common usage is to refer to the execution of AI models at the edge.

Though, they define it as the efficient use of data in an edge-cloud cooperative manner, where inference and training can happen across all devices. We also believe in that vision and showcase in our use case that data from different domains can be used to create new and intelligent applications. Zhou et al. [5] also introduce a six-level rating that specifies where applications are executed. We briefly highlight the main differences, and advise to read a detailed explanation in [5]:

- Cloud Intelligence: Training and inference on the cloud
- *Level 1* Cloud-edge co-inference:
  Training: Cloud - Inference: Edge-Cloud
- *Level 2* In-edge co-inference:
  Training: Cloud - Inference: Edge
- *Level 3* On-device inference:
  Training: Cloud - Inference: Device (no offloading)
- *Level 4* Cloud-edge co-training:
  Training: Edge-Cloud - Inference: Edge-Cloud
- *Level 5* All in-edge:
  Training: Edge Inference: Edge
- *Level 6* All on-device (no offloading):
  Training: Device - Inference: Device

Because EC contains resource-constrained devices, some tasks must be offloaded to other nodes. Other surveys have comprehensively shown open issues we encounter in EI [4], [12]. Similar studies have been published that investigated the complete food supply chain. Pang et al. [13] showcase how IoT can influence food supply chains. They focus on conducting surveys on the importance of IoT applications with experts (i.e., shelf life prediction) and describe in detail the technical realization of those sensor networks, including sensor data fusion. In [14], the authors present a serverless platform for Edge Intelligence applications. Their focus lies in discussing the technical details but also introduce the problem that arises from context-aware systems. The work is complementary to ours, as they propose a programming model and low level details about the execution platform. Zhang et al. [15] present an open framework that focuses on the technical implementation of EI applications at the edge. An important issue is the large size of AI models and the resulting mismatch with resource-constrained devices. They propose a systematic description of EI algorithms, a RESTful API to expose EI applications and outline usage with four use cases. We look at Edge Intelligence from a higher level, by describing a plausible business, how it can benefit from EI and that not only the computing infrastructure is heterogeneous. Further, our platform focuses on the democratization of AI, that will push development and use to a broader public. The work of Rausch et al. [6] resemble our approach but differs in the approach, in that ours is based on business case study which highlights the importance of all these different applications to a singe stakeholder.

III. Motivating Use Case

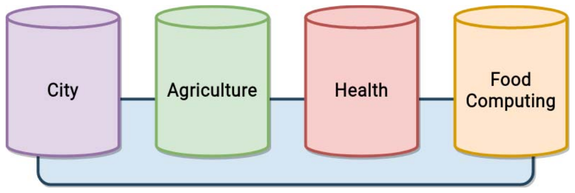

Fig. 1. Crossing domain boundaries

Our motivating use case presents possible EI applications that can be developed and deployed with our proposed end to end platform. The use cases are settled in the context of Smart Cities [16], Smart Agriculture [17], Smart Retail [18], Smart Health [19] and Food Computing [20], depicted in Figure 1. We aim to showcase the potential of EI to help a business that produces vegetables. Each application is briefly described by focusing on necessary data, the possible EI levels, which domains are involved and which elasticity requirements are important. Table I summarizes our findings.

Consider an international company, set up over different countries, that (1) operates a commercially agricultural business and (2) sells the produces in stores. Whereas products can originate from the local area or foreign countries and the business is in direct contact with the farmers. The main revenue stems from selling food to individuals in shops with accompanying restaurants, making the business' goal to adapt their offering to the cities. We briefly highlight the business' lifecycle, and afterwards showcase useful applications that can be deployed on an EIaaS platform. We consider a simplified process that is split into two parts: farm and shop. Farmers want to efficiently plant seeds, take care of plants and yield high quality crops. The shop gets them delivered, places them in-store where customers buy them, either the raw produce or as dish served in the in-store located restaurant. Based on analysis of sales, business and farmers can adopt their strategies with help of a Recommender System. Figure 2 shows each step and indicates the feedback loop. After outlining the business' internal process, we describe for each step plausible applications that could help reduce cost and improve quality.

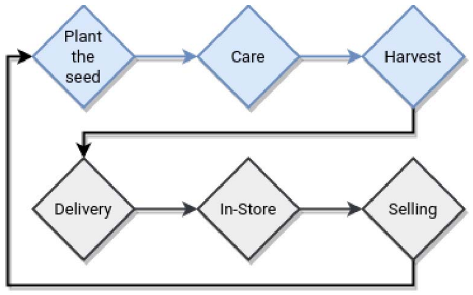

Fig. 2. Steps in plant farming chain

A. Farm

1) Plant: First, farmers can use AI to map the soil to assess chemical characteristics. Aitkenhead et al. [21] trained a neural network that can predict soil characteristics to estimate fertility. Their approach appears optimal for cheap and rapid field assessment using colour values from digital photographs [21]. A crowdsourced labelling process can prove to be efficient and therefore inference and training at the edge are recommended. Further, elasticity requirements will target almost exclusively the reduction of cost, as analysis does not have real time requirements and only serves as a cheap assessment without focusing on high accuracy.

2) Care: During the growing phase of plants, farmers can take advantage of automatic discrimination between crop and weed based on images [22], [23], plan irrigation with short-term weather forecast [3] and perform vision-based disease detection [24], [25]. Crop and weed prediction has to perform in real-time to allow tractor mounted spraying equipment efficiently target weed patches between crop rows. Both, crop/weed discrimination and plant disease detection, can profit from localized fine-tuning due to the large amount of different species [26]. Climate and soil are vastly different across the globe directly affecting watering cycles and fertilization, making fine tuning at the edge a viable solution to guarantee high accuracy results [3].

3) Harvest: Multiple vision-based systems can be deployed to infer fruit size and quantity and help farmers during harvest to predict the quality [27], [28]. Yield prediction of winter wheat has proven to be sensitive towards location and input's time window [29]. Han et al. [29] propose a model that predicts yield of winter wheat in china. Their model uses remote sensing (i.e., vegetation index), climate (i.e., temperature), soil (i.e., soil properties) as well as historic yield data and crop maps. Results show that the local environment is highly relevant for the model and bigger surroundings lead to worse accuracy. Halstead et al. [28] have stated that their system is trained with near-maturity peppers and fails to accurately predict ripeness of juvenile fruits. Their two-stage approach of first detecting the fruits and afterwards estimating the ripeness, shows that the second stage may have to be fine tuned at the edge, to guarantee high accuracy by using multiple models.

B. City

1) Transportation: After the harvest, produces must be safely transported to the shops and therefore can benefit from crowdsourced road condition assessment, smartphones measure the movement via accelerometer and gyroscope [30], [31]. Not only does it lead to increased traffic safety, it also avoids any unnecessary spoilage caused by continuous vibrations and high force shocks [13]. In road sensing the quality is important, as trips can be planned ahead and therefore do not require low latency. We think that a localized fine tuning can yield higher accuracy due to differences of common vehicle characterizations (i.e., suspensions) that may lead to different accelerometer and gyroscope readings on the same road. Further, AR-based helper systems, that receive results from nearby traffic cameras, can display nearby traffic participants in real time [32]. The results contain bounding boxes of inferred objects in the camera's view. Locality-awareness is of utmost importance to guarantee relevant results and ultra low latency, required to continuously provide the AR system with data. Consider that throughout the city cameras are deployed and all of them are running object detection models that emit bounding boxes representing participants. Drivers want and need only those that are in near proximity, the system has to automatically adept the transferred data based on the users location. While accurate inference is important, performance must also be considered to make the application usable. Training can be done in the cloud using large datasets, but inference must happen at the edge due to latency requirements.

2) In-Store: Upon delivery, produces are put into shelves or prepared as meals. Customers should be engaged and incentivized to buy products during their stay in the local shop. Due to the emergence of Virtual and Augmented Reality [32], we assume customers are wearing smart glasses to get additional information while strolling through to store.

Smart glasses record the user's view with up to 30 frames per second, which act as input for an inference pipeline. This pipeline identifies objects, selects products and classifies them [33]. Based on the resulting class of food, additional information of the user's interest is shown. Information may include:: detailed nutritional information, possible recipes, ecological impact (i.e., consider the transport of food from foreign countries), etc. The application is context-aware and can use personal data to provide relevant information. Multiple works have shown promising results in recognizing food and dishes [34], [35], [36], [37]. Xu et al. [33] have added geo-location information to the model, to better differentiate between dishes that look the same and stem from different cuisines. This approach seems plausible in the context of EI, where a general food recognition model can be trained in the cloud and fine tuned accordingly in different restaurants and/or markets. Further, processing the captured video feed of smart wearables must adhere to stringent latency requirements and may even be subject to local policies protecting people's identities (i.e., GDPR). Processing can be done either on-device (i.e., smartphone) or nearby edge devices and sophisticated anonymization software can act as pre-processing to obey laws [38]. Based on personal preferences, applications may recommend food during the customer's visit [39], [40]. Smart health wearables may provide applications with contextual data to react accurately to different situations. These applications process highly sensitive data (i.e., allergens, dietary, vital data) and therefore training on-device is required. This restriction can build the important trust between user and application [41].

3) Sale: After users make their choice, smart checkouts using deep learning based recognition software can help shops ease the workload on workers, but need to be accurate and fast at the same time [42]. Which makes us believe that a cloud-based training and edge inference are suitable. Context-aware recommender systems can learn user preferences based on previous purchases [43]. Privacy is important for this application type, and therefore we recommend that confidential tasks are done on-device.

4) Feedback: The last use case revolves around the idea of creating a Recommender System that can help the business and farmers to effectively recommend the next crop. We imagine that this system can build on nearly all previously mentioned data sources. For example, historical sale reports can highlight seasonal patterns, the local fields can be analysed for soil mapping purposes as well as weather data influences feasible crops. Analysis of personal data (i.e., dietary choices, allergens, preferences, etc.) can foster the system with useful information regarding the overall consumption behaviour of cities, or even districts. Besides data about the citizens of Smart Cities, environmental data of the surroundings can further influence recommendations. In areas with high pollution, trees can help mitigate the issue and produce fresh oxygen. To the best of our knowledge, no work has been done in this direction that combines Smart City and Smart Agriculture to create a feedback loop between the production and consumption of vegetables.

We observe that the applications differ widely in terms of input data, elastic requirements and EI levels. Data from the domains of Smart City, Agriculture and Health, as well as Food computing were used. We process various types of data: climate, soil, high dimensional (i.e., images), plant, health and personal data. While some models may be trained in the cloud and are deployed at the edge for inference, others have to be aware of contextual information and therefore are fine tuned at the edge. Privacy-related concerns make on-device inference and training necessary, as well as the reduction of performance and the conquering of large amounts of data. Elasticity requirements differ in terms of quality, performance, privacy, and should balance cost. Further, we identify issues that relate to the tasks we discuss in the next section: (1) addressability, (2) sources, (3) security, (4) coordination, (5) performance and (6) context.

IV. Tasks

In the following, we categorize operational tasks that our framework will focus on. Our opinionated view on application requirements allows us to define general as well as highly contextual goals.

We consider the four main tasks that appear in Edge Intelligence: sensing, preprocessing, training and inference. This classification is based on recent research that identified different layers and tasks [15], [6]. We highlight for each task different aspects and key requirements encountered in our use case study.

A. Sensing

Our use cases show that physical sensors can be deployed anywhere and measure different types of data. Table II shows few of them that can be used to achieve the presented applications. For each sensor we also describe the collected data types and which domain they belong to. Additionally, other studies have investigated the heterogeneity in sensors and emitted data [45], [46], [47], [48], clearly showing the need for a unified interface. The heterogeneity stems from different types of velocity, volume and ownership, as depicted in Table II. Therefore, we think a layer of abstraction is necessary to make this data accessible. The layer should provide ways of selecting, merging and aggregating data from different sources while being context-aware.

Next, we highlight requirements that stem from our motivating use case.

1) Addressability: The use cases made it clear that efficient querying for sensor data is necessary and needs to handle context. Nearby traffic participant detection is practically unusable without the knowledge of the users whereabouts. Users must be able to either send information to the system that it only sends relevant resources, or must employ a fine-grained topic subscription to enable these use cases. While in this application users can specify exactly which sensors are needed (i.e., using geospatial queries to select all cameras in 100m radius), others require the system to understand much more semantically queries. Consider personal food recommendations that requires highly contextual data to train an accurate model. How should developers approach this daunting task of selecting all sources, containing smartphone sensors, personal health records, etc.? Therefore, being able to easily address, select, merge and process data from sensors is a very important task. Establishing trust to users (i.e., guaranteeing on-device processing) can be a key enabler for creating a homogeneous platform, providing access to sensors from external providers. For example, imagine we want to create a model that relates temperature and crop growth rate in the area surrounding Vienna. The system must combine temperature sensors with crop sensors in vicinity. A system that fails to identify relationships between sensors will yield useless models. Therefore, sensors need to be tagged with context such that we can query and group them.

TABLE I
POSSIBLE APPLICATIONS IN THE SUPPLY CHAIN

Step	Application	Data	EI Level	Domains	Elasticity
Plant the seed	Soil Mapping [44]	Topographic Climatic Vegetative indices	Level 4	Smart Agriculture	Quality, Context
Care	Crop/Weed discrimination [22], [23] Irrigation Scheduling [3] Disease Detection [24]	Weather RGB, Spectral images	Level 1 & 5	Climate Smart Agriculture	Cost, Quality, Privacy
Harvest	Fruit Size Estimation [27], Fruit Quantity Estimation [28] , Yield Prediction [29]	RGB(-d) images Remote Sensing Climate, Soil Yield, Crop map	Level 4	Smart Agriculture, Climate, Food Computing	Quality Performance
Transportation	Traffic participant detection [6] Road condition assessment [30], [31]	Images, Orientation Angular Velocity	Level 3 & 4	Smart Traffic	Safety
In-Store	Food Classification [33] Food Recommendation [1], [40] Face anonymization [38]	Images, Health Personal preferences	Level 3 & 4	Food computing, Smart Health	Privacy, Quality, Context, Performance
Sale	Intelligent checkout [42] Personalized recommender [43]	Images, Location Temporal, Consumer Behavior	Level 2 & 6	Smart Retail	Privacy, Context Performance
Feedback	Produce Recommendation	Sales, Social context Location	Level 4	Smart Business	Privacy, Quality Context

2) Sources: Different types of sources have been proposed that can act as sensors. They are not restricted to be of physical nature (i.e., temperature sensor), but can also be virtual or logical [49]. All three groups are used throughout our use cases. Table II displays sensors that were used in the presented applications. Virtual sensors are characterized by gaining information from software. A smart traffic light, using cameras, can emit bounding boxes for detected objects. Other applications can use these bounding boxes, i.e., to display them in a AR headset [32]. Another plausible and important virtual sensor is anomaly detection. Anomaly detection can be used in wireless sensor networks, where we want to be notified in case of sensor failure [50]. Logical ones combine different sensors to emit aggregated data. A personalized recommender may make use of the user's location (measured by GPS sensor) and takes previous shopping behavior into consideration. Therefore, the platform must support not only physical sensors but has to be flexible in regards to the definition of what constitutes a source. Additionally, we roughly estimate the velocity and volume for each sensor and the ownership. It is important to highlight that sensors are not only heterogeneous in terms of data type and other factors increase the complexity of data fusion.

B. Preprocessing

After defining the sources, AI applications typically perform some preprocessing on data and build the first step accomplished on the computing infrastructure. Either for training purposes (i.e., creating batches), inference (i.e., image scaling) or for exploratory analysis (i.e., aggregation). Gravina et al. [51] argue that a seperated preprocessing can benefit application development because developers can focus on the functionality while skipping carefully selecting and filtering data from the sensor. Edge Computing provides the capability of preprocessing large amounts of data emitted by sensors in a timely and efficient manner (i.e., by placing applications near the origin). Through the ability of placing applications at the edge, platform providers can guarantee data security and privacy preservation by leveraging light-weight methods to protect data, identity and location [52]. By combining local processing of sensitive data and pseudo-anonymization, platforms are able obey to local laws and make guarantees about safety for users. This requirement differs across application types, but the presented applications that process highly sensitive and personal data. In the following we briefly discuss these security concerns.

1) Security: Our use case study showcased multiple applications that may act on private data (i.e., personalized food recommendation). Especially Smart Health applications fundamentally require personal data and therefore need to protect the user's privacy. It is important to act only after users give consent and are informed of the data being processed [53], [41]. For example, our proposed food recommendation application, that processes data about the user's surroundings, activity, dietary preferences, allergens, body weight, etc., will probably not be adopted by users if they do not trust the platform. A survey in the field of Smart Health, conducted by Li et al. [54], has shown that: users adopt healthcare wearable through a subjective risk-benefit assessment, whereas the perceived privacy risk is a combination of different factors (i.e., legislative protection), and the benefit is subject to perceived informativeness and functional congruence. Therefore, the platform and applications have to be designed with privacy related issues in mind to be accepted and adopted by the general public.

TABLE II
POSSIBLE SENSORS AND CHARACTERISTICS

	Sensor	Data	Velocity	Ownership	Volume
Smart Agriculture	Crop Location	Tags the location of crops	Slow	Private	Low
	Temperature	Temperature of location	Normal	Public	Low
	Humidity	Humidity of location	Normal	Public	Low
	Camera	Identified objects on-premise	Ultra fast	Private	Low
	Spectral Camera	Fruit ripeness detection	Fast	Private	Normal
	Camera	Raw video feed	Ultra fast	Private	Very High
Smart Health & Food Computing	Smartphone	User location	Fast	Private	Low
	Smart glasses	User's Field of View	Ultra fast	Private	Very High
	User	Dietary	Slow	Private	Low
	User	Allergens	Slow	Private	Low
	User	Age	Slow	Private	Low
	User	Residence	Slow	Private	Low
	BAN	Activity Recognition	Fast	Private	Low
Smart City	Camera	Traffic Participants	Ultra fast	Semi-Public	Low
	Camera	Raw video feed	Ultra fast	Semi-Public	Very High
	Temperature	Outdoor temperature of area	Slow	Public	Low
	Barometer	Air pressure	Slow	Public	Low

C. Training

Training at the edge has several advantages: enabling to learn on raw sensor data (i.e., no aggregation necessary due to possible on-device training), reduction of bandwidth pressure, and privacy preservation. But training is an expensive computational task and can require large amounts of computing resources. While embedded devices, equipped with hardware accelerators, are commercially available (i.e., Nvidia Jetson devices [55]), it remains a challenging task to fully utilize all devices. Difficulties stem from the storage and computational demanding neural networks which pose a problem with regards to resource-constrained devices. Further, privacy preservation is important for some use cases and training must happen locally. Federated Learning tackles these issues and represents a distributed and privacy-preserving model training approach that can even be applied on resource-constrained devices (i.e., smartphones) [56]. Other works [4], [12] have already highlighted several issues concerning Federated Learning (i.e., worker selection) and due to our focus on investigating context-awareness, we disregard these issues for a moment and focus on the coordination obstacles that are specific to enable localized fine-tuned models.

1) Coordination: Our use case has shown the potential of training multiple models based on the context of deployment. For example, personalized food recommendation depends heavily on the user and also has high security requirements to protect the privacy. Food classification has also shown potential to benefit from context-aware fine tuning due to visual similarity of dishes [33]. Further, coordination can also include the task of monitoring AI model performances and in case of a decline new training rounds have to initiated. Model degradation can happen due to new data which may require re-training to perform better [57]. Therefore, the platform has to intelligently coordinate training not only based on context and possibly training of multiple models but also has to monitor the performance. Fine-tuning may not only improve accuracy for specific tasks, but also training with unlabeled data can increase robustness of the model and reduces the surface of attack vectors [58]. Due to the abundance of available information, unlabeled data can be easily retrieved and strengthens the models.

D. Inference

While training AI models requires sophisticated strategies to enable training on resource-constrained devices and worker selection. We argue that this task is not subject to stringent latency requirements that we encounter in inference applications. Our use case has shown applications that require ultra low latency and performance is of utmost importance. Besides performance, context-awareness can also introduce new issues. Consider our fine tuned models that are trained with a specific context in mind. This raises the problem of choosing the right model depending on the users' context. Both issues are addressed in the following and conclude the section.

1) Performance: Due to the high heterogeneity in terms of performance, Edge Computing infrastructures require intelligent scheduling mechanisms to reason about placements. Though, performance differences are not the only issue and performance degradation caused by multi-tenancy is a serious issue that needs focus when developing a EIaaS platform. In general, multi-tenancy issues can rise issues in two ways: (1) access to exclusive resources, and (2) performance interference. We identify that some resources may be exclusive to one application, but in other cases multi-tenancy situations may occur by placing multiple applications on one node. Multi-tenancy can already lead in cloud settings to performance issues [59], and we encounter the same problem on resource-constrained devices. The platform requires an intelligent scheduler that is aware of those issues and prevents them from happening. A problem that we also face when considering performance and resource-constrained devices is the mismatch between large AI models and resource-constrained

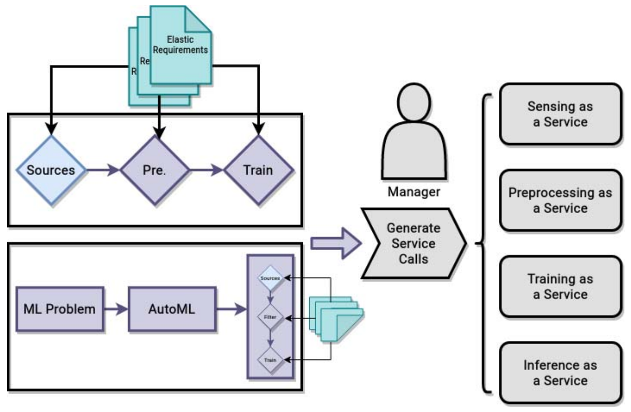

Fig. 3. End-to-End platform

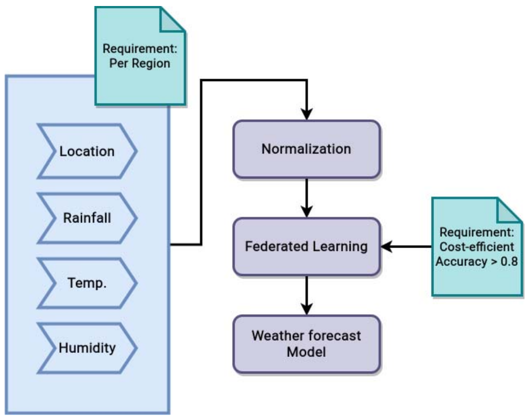

Fig. 4. Training pipeline

devices [15]. Techniques are offered to mitigate these issues (i.e., model splitting with computational offloading) but it remains open who will be responsible for this to decide. In certain circumstances, some edge devices will be capable to run the full model, avoiding issues that come with offloading. Therefore, the platform (or user) have to decide which method is most suitable.

2) Context: As already hinted in our use case study, fine-tuning models for specific contexts is favorable in contrast to having only a single all-knowing model. This stems from the facts that models may not generalize well and training with unrelated data may lead to a decline in performance, and also use cases require multiple models due to intrinsic properties of the targeted problem (i.e., similarity in dish pictures). Additionally, personalized models also have to be maintained and associated with their corresponding user. Therefore, users need to be able to specify their current context, such that the platform can choose the right model.

V. Platform

We now present our envisioned EIaaS platform that addresses the previously described challenges. It is opinionated towards Edge Intelligence applications and focuses on providing an accessible interface, to push the democratization of AI further. While the proposal is based on the assumption of a working Edge Computing infrastructure, we acknowledge that many problems are yet unsolved and, therefore, showcase ideas from which general purpose Edge Computing infrastructures can benefit. First, we depict the platform's architecture and on which deployment methods we would build on. Secondly, we present our vision of the platform's interface, how users interact with it, and can influence the application's behavior. Afterwards, we go deeper into technical details and define a plausible scheduler, based on a greedy multi-criteria decision algorithm, enforcing user-defined requirements.

A. Architecture

Operationalizing AI applications is an emerging topic and focuses on automating the lifecycle of AI models. Companies, like Google [11] and Amazon [10], [60], have released platforms that offer end-to-end development and lifecycle management for AI. Besides commercial offerings, these systems have also been investigated in open source and academic communities [61], [62], [63], [14]. The general approach, MLOps, is based on the DevOps paradigm [64] that automates the lifecycle of application development (i.e., packaging, testing, deploying, versioning, etc.). In the same manner, AI applications are managed using pipelines, which model the lifecycle of AI models as a directed acyclic graph. In addition to the discussed tasks in AI (i.e., training), continuous monitoring is important to identify concept drifts, which negatively impact a model's accuracy [65]. MLOps platforms thrive to automate the whole AI lifecycle and are therefore suitable to build the base of our platform. Pipelines can be declared using either configuration files [61] or in a UI [11]. The important thing is, that the underlying platform receives a description of the pipeline and can compile this into actual service calls. This gives us the flexibility to take user defined requirements into account.

B. Platform workflow

By assuming the underlying platform is similar to systems such as Google's VertexAI [11], KubeFlow [66] or ModelOps [61], we describe now how user interact with the platform. Specifically, our focus lies in democratizing AI and giving users the ability to add requirements, as well as context. Figure 3 depicts a high level interaction possibility. By leveraging AutoML techniques, user can define the underlying problem through selecting different sources that build the input and output space of the problem. AutoML techniques help democratize the usage and have gained notable traction over the years [6], [67]. While we think that AutoML can be beneficial, users should also be able to define pipelines manually. This description gets translated into a pipeline, to which users can add elastic requirements. Afterwards, the pipeline is transferred to the manager identity, who takes care of the continuous execution. Pipeline steps are translated into service calls, which are available for each task. Users can add to each step in the pipeline individual elastic requirements that influence the applications behavior. The requirements add context to the operations and allows the system to reason about when it comes to operational strategies (i.e., placement strategy). Figures 4 & 5 showcase the definition of two pipelines. Figure 4 shows a training pipeline for a short-term weather forecast model [3]. In the first step, users have to define the data sources, which in this case are physical sensors that combine location and weather data. The associated requirement *Per Region* indicates the platform to group the sensor data and create multiple models. Afterwards, we pre process the data and can then train on it. The model is either deployed automatically and/or stored for future applications. In our case, we specify the training to reach an accuracy of 80% and want the platform to perform this training in a cost-efficient manner. These elastic requirements are directly related, as federated learning utilizes a worker concept to train data which consequentially has a big impact on the resulting cost. Executing this task will depend on many factors that all affect the cost and effectiveness. For example, the resulting accuracy will depend on the number of training rounds, which determine the cost for the training. The manager has to be aware of this relationship and must balance the execution between these two goals. Figure 5 shows a dedicated pipeline for inference consisting of different models that fulfill a certain task. We depict the use case of recommending users food they see while wearing Smart Glasses. Recent work in optimizing these kind of pipelines [68] and the re-usability of models, makes us believe that the separation is beneficial for all involved parties. Elastic requirements aid the deployment of offloading applications by stating which step has to be executed on-device. Efficient offloading is necessary to make model splitting techniques feasible. This knowledge can further be used of the underlying services to possibly place interdependent functions in near proximity, to avoid network latencies. Requirements include the processing of sensitive data on-device and imposes a hard constraint towards the round trip time (RTT). The RTT is picked according to the assumption of sending 30 pictures per second to create a smooth UX. Crankshaw et al. [68] present their system that takes tight tail latency constraints into account while considering different hardware accelerators, but are situated in cloud computing.

C. Technical details

After presenting our vision to develop EI applications, we now turn over to describing possible implementation strategies. We consider the necessary technologies to be different from the sensing task in contrast to the others. Therefore, the platform uses a message-oriented publish/subscribe layer to send sensor data to processing applications. Subscribers can receive messages via topics from various nodes by specifying certain identifiers. Further, dynamic message brokers have been proposed that can scale in a location-aware way, such that messages get routed over a nearby message broker to reduce latency [69]. Complementary to Rausch et al. [14], we

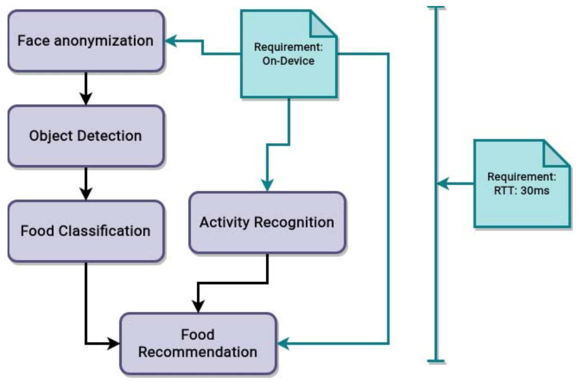

Fig. 5. Inference pipeline

envision the remaining services to be based on Serverless Edge Computing, whereas each step in the pipeline is implemented as a function [70]. Therefore, containers include a single stateless application that exposes a single function over the network (usually HTTP). Kubernetes [71], the underlying container orchestration service of Kubeflow, utilizes a greedy multi-criteria decision algorithm to decide which node is the most suitable one for a given function. Currently, the input of this algorithm consists of a given function and a node, which (1) filters infeasible nodes (i.e., node does not have required accelerator) and (2) calculates a score based on an extensible list of scoring functions. This approach has been proven to work well on different infrastructures and AI applications [72]. The scoring approach offers a highly flexible way of influencing scheduling decisions and can be arbitrary complex. A limitation of the current Kubernetes ecosystem is, that we cannot easily bundle multiple function container that act equally but have small differences. Users should be able to upload multiple implementations of one function that are equal in functionality. Further, users can add information to each function implementations, such that the service can reason about them and intelligently select the optimal one. Consider our federated learning requirements, in which the user expresses cost-efficiency. While neural network applications greatly benefit from using GPUs, the operation may cost more than using the CPU. Therefore, in some cases the scheduler may prefer the CPU computing platform. Through the combination of an intelligent manager, and the underlying scoring functions, we can create a system that can handle EI applications. For example, locality-awareness is of utmost importance. Consider a platform that manages multiple smart city deployments. Increased usage in one city does not affect the deployed applications in another and scaling must be able to identify this aspect and make the scheduling process aware of this. This problem can be solved by the following steps: (1) develop a scaler that can differentiate between the two cities, i.e., identify the need for more function instances in one city. (2) integrate an intermediary component to which scaler and scheduler can talk. The scaler would need to put additional information concerning the location into this component. (3) scoring functions need to retrieve any additional info, and therefore can be location-aware. To the best of our knowledge, Kubernetes currently scales and schedules independently and scheduling decisions are unaware of scaling reasons. If there is in fact a way to pass messages between these two components, our approach would still be plausible by removing the external component. The important take-away is that the scoring approach is highly extensible to vastly different circumstances that we encounter in EI. Adding to the problem of location-awareness, load balancers have to be distributed and scaled with location in mind. Considering that ultra low latency requirements can only be satisfied in case the route to the cloud can be skipped.

VI. Limitations & future work

A. Theoretical approach

An important limitation of our work is the theoretical approach. While we base our user case, the requirements, and the platform on extensive literature research, we have to mention that this work does not guarantee that the system has to work as we described. Though, with the current advances in all discussed topics and the existing research we base our system on, we are confident that such systems will exist at some point. The concept of context-aware systems exist for decades [73], and with the current development of *Smart-** concepts, Edge Computing and Edge Intelligence, these ideas may become soon reality.

B. Centralized vs. Decentralized

Already briefly discussed in Section V-C, service components may need to be decentralized. We consider the load balancer to be the first component that has to be decentralized, otherwise we can not fulfill ultra low latency requirements. It is unclear if other components have to be decentralized too, i.e., scheduler and scaler. Rausch et al. [14] have investigated the throughput of the Kubernetes scheduler and concluded that a monolithic approach is not feasible. While they do advocate the use of disaggregated schedulers, a decentralized one can also aid scalability considering geo-distributed deployments. Caveats of decentralization are not only related to the management overhead that comes with such systems, but also security issues arise. Blockchain technologies can help mitigate these [74].

C. Governance

Currently it is not clear who will own Edge Computing infrastructures and how access will be managed [6]. This issue is especially concerning for the sensing task. Questions about data ownership will have to be answered and if publish/subscribe system are enough in the face of multiple parties involved (i.e., government, businesses). Therefore, our platform is based on the Sensing as a Service [75], [76] concept and leaves the technical implementation open for now, while allowing us to formulate requirements.

VII. Conclusion

Holistic Edge Intelligence as a Service platforms are yet subject for exploration and research. Currently, we are facing the emergence of new paradigms that range from infrastructure (Edge Computing), application development (MLOps), to applications (Edge Intelligence). The paradigms move resources and applications towards the edge of the network and solve problems related to the increasing amount of data produced, development of ultra low latency applications, privacy concerns, and democratization. Edge Computing can be seen as a geo-distributed cluster with high heterogeneity, devices range from Raspberry Pi to fully-fledged cloud servers. MLOps promises to open up the space of developing robust AI applications to the broader public by declaring pipelines and use of AutoML techniques. Edge Intelligence is currently being investigated and its limits are not yet clear. Our work presents a cross-domain reaching use case study that defines in great detail different applications settled in vastly different domains. We combine Smart City, Health, Agriculture, and Food Computing to showcase possible intersecting use cases that make use of heterogeneous sensors. Different elastic requirements are outlined that have to be defined for each application and additionally identify their EI level. After explaining the key characteristic for each device, we describe the four main tasks encountered in Edge Intelligence and identify challenges that have to be tackled to make full use of its potential. Our proposed platform builds on state-of-the-art MLOps systems and tackles each identified requirement. We highlight the platform workflow with two different applications and show that due to the nature of AI pipelines our system can reach a broad audience and therefore push the democratization of AI forward. Elastic requirements are first-class citizens and can augment the reasoning allowed by functions, that represent our unit of deployment. Further, a possible technical implementation is outlined and has been proven by others to show the possible potential of an online MCDM scheduler. We are aware that our work is theoretical and work has to be done to realize this proposal. Especially considering that issues often arise during actual development. Though, due to our close connection with proven techniques, we are confident that EIaaS platforms can be built on our proposed concept.

Acknowledgment

The authors would like to thank Thomas Rausch for his expertise and invaluable comments.

References

[1] L. Yang, C.-K. Hsieh, H. Yang, J. P. Pollak, N. Dell, S. Belongie, C. Cole, and D. Estrin, "Yum-me: a personalized nutrient-based meal recommender system," *ACM Transactions on Information Systems (TOIS)*, vol. 36, no. 1, pp. 1–31, 2017.

[2] 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.

[3] J. Cao, J. Tan, Y. Cui, and Y. Luo, "Irrigation scheduling of paddy rice using short-term weather forecast data," *Agricultural water management*, vol. 213, pp. 714–723, 2019.

[4] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: the confluence of edge computing and artificial intelligence," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7457–7469, 2020.

[5] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.

[6] T. Rausch and S. Dustdar, "Edge intelligence: The convergence of humans, things, and ai," in *2019 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 2019, pp. 86–96.

[7] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[8] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, 2016.

[9] G. A. Montes and B. Goertzel, "Distributed, decentralized, and democratized artificial intelligence," *Technological Forecasting and Social Change*, vol. 141, pp. 354–358, 2019.

[10] Amazon, "Amazon sagemaker." [Online]. Available: https://aws.amazon.com/sagemaker/

[11] Google, "Vertex ai." [Online]. Available: https://cloud.google.com/vertex-ai

[12] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2020.

[13] Z. Pang, Q. Chen, W. Han, and L. Zheng, "Value-centric design of the internet-of-things solution for food supply chain: Value creation, sensor portfolio and information fusion," *Information Systems Frontiers*, vol. 17, no. 2, pp. 289–319, 2015.

[14] T. Rausch, W. Hummer, V. Muthusamy, A. Rashed, and S. Dustdar, "Towards a serverless platform for edge {AI}," in *2nd {USENIX} Workshop on Hot Topics in Edge Computing (HotEdge 19)*, 2019.

[15] X. Zhang, Y. Wang, S. Lu, L. Liu, W. Shi *et al.*, "Openei: An open framework for edge intelligence," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 1840–1851.

[16] V. Albino, U. Berardi, and R. M. Dangelico, "Smart cities: Definitions, dimensions, performance, and initiatives," *Journal of urban technology*, vol. 22, no. 1, pp. 3–21, 2015.

[17] D. C. Rose and J. Chilvers, "Agriculture 4.0: Broadening responsible innovation in an era of smart farming," *Frontiers in Sustainable Food Systems*, vol. 2, p. 87, 2018.

[18] S. G. Dacko, "Enabling smart retail settings via mobile augmented reality shopping apps," *Technological Forecasting and Social Change*, vol. 124, pp. 243–256, 2017.

[19] A. Solanas, C. Patsakis, M. Conti, I. S. Vlachos, V. Ramos, F. Falcone, O. Postolache, P. A. Pérez-Martínez, R. Di Pietro, D. N. Perrea *et al.*, "Smart health: A context-aware health paradigm within smart cities," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 74–81, 2014.

[20] W. Min, S. Jiang, and R. Jain, "Food recommendation: Framework, existing solutions, and challenges," *IEEE Transactions on Multimedia*, vol. 22, no. 10, pp. 2659–2671, 2019.

[21] M. Aitkenhead, M. Coull, W. Towers, G. Hudson, and H. Black, "Prediction of soil characteristics and colour using data from the national soils inventory of scotland," *Geoderma*, vol. 200, pp. 99–107, 2013.

[22] W. Guo, U. K. Rage, and S. Ninomiya, "Illumination invariant segmentation of vegetation for time series wheat images based on decision tree model," *Computers and electronics in agriculture*, vol. 96, pp. 58–66, 2013.

[23] M. Aitkenhead, I. Dalgetty, C. Mullins, A. J. S. McDonald, and N. J. C. Strachan, "Weed and crop discrimination using image analysis and artificial intelligence methods," *Computers and electronics in Agriculture*, vol. 39, no. 3, pp. 157–171, 2003.

[24] D. Moshou, C. Bravo, J. West, S. Wahlen, A. McCartney, and H. Ramon, "Automatic detection of 'yellow rust'in wheat using reflectance measurements and neural networks," *Computers and electronics in agriculture*, vol. 44, no. 3, pp. 173–188, 2004.

[25] D. Moshou, C. Bravo, R. Oberti, J. West, L. Bodria, A. McCartney, and H. Ramon, "Plant disease detection based on data fusion of hyper-spectral and multi-spectral fluorescence imaging using kohonen maps," *Real-Time Imaging*, vol. 11, no. 2, pp. 75–83, 2005.

[26] X. P. Burgos-Artizzu, A. Ribeiro, M. Guijarro, and G. Pajares, "Real-time image processing for crop/weed discrimination in maize fields," *Computers and Electronics in Agriculture*, vol. 75, no. 2, pp. 337–346, 2011.

[27] Z. Wang, K. B. Walsh, and B. Verma, "On-tree mango fruit size estimation using rgb-d images," *Sensors*, vol. 17, no. 12, p. 2738, 2017.

[28] M. Halstead, C. McCool, S. Denman, T. Perez, and C. Fookes, "Fruit quantity and ripeness estimation using a robotic vision system," *IEEE robotics and automation LETTERS*, vol. 3, no. 4, pp. 2995–3002, 2018.

[29] J. Han, Z. Zhang, J. Cao, Y. Luo, L. Zhang, Z. Li, and J. Zhang, "Prediction of winter wheat yield based on multi-source data and machine learning in china," *Remote Sensing*, vol. 12, no. 2, p. 236, 2020.

[30] A. Allouch, A. Koubâa, T. Abbes, and A. Ammar, "Roadsense: Smartphone application to estimate road conditions using accelerometer and gyroscope," *IEEE Sensors Journal*, vol. 17, no. 13, pp. 4231–4238, 2017.

[31] F. Seraj, B. J. van der Zwaag, A. Dilo, T. Luarasi, and P. Havinga, "Roads: A road pavement monitoring system for anomaly detection using smart phones," in *Big data analytics in the social and ubiquitous context*. Springer, 2015, pp. 128–146.

[32] T. Rausch, W. Hummer, C. Stippel, S. Vasiljevic, C. Elvezio, S. Dustdar, and K. Krösl, "Towards a platform for smart city-scale cognitive assistance applications," in *2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, 2021.

[33] R. Xu, L. Herranz, S. Jiang, S. Wang, X. Song, and R. Jain, "Geolocalized modeling for dish recognition," *IEEE transactions on multimedia*, vol. 17, no. 8, pp. 1187–1199, 2015.

[34] V. Bettadapura, E. Thomaz, A. Parnami, G. D. Abowd, and I. Essa, "Leveraging context to support automated food recognition in restaurants," in *2015 IEEE Winter Conference on Applications of Computer Vision*. IEEE, 2015, pp. 580–587.

[35] Y. Kawano and K. Yanai, "Real-time mobile food recognition system," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2013, pp. 1–7.

[36] G. Ciocca, P. Napoletano, and R. Schettini, "Food recognition: a new dataset, experiments, and results," *IEEE journal of biomedical and health informatics*, vol. 21, no. 3, pp. 588–598, 2016.

[37] Y. Kawano and K. Yanai, "Foodcam: A real-time food recognition system on a smartphone," *Multimedia Tools and Applications*, vol. 74, no. 14, pp. 5263–5287, 2015.

[38] H. Hukkelås, R. Mester, and F. Lindseth, "Deepprivacy: A generative adversarial network for face anonymization," in *International Symposium on Visual Computing*. Springer, 2019, pp. 565–578.

[39] G. Agapito, B. Calabrese, P. H. Guzzi, M. Cannataro, M. Simeoni, I. Caré, T. Lamprinoudi, G. Fuiano, and A. Pujia, "Dietos: A recommender system for adaptive diet monitoring and personalized food suggestion," in *2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 2016, pp. 1–8.

[40] D. Bianchini, V. De Antonellis, N. De Franceschi, and M. Melchiori, "Prefer: A prescription-based food recommender system," *Computer Standards & Interfaces*, vol. 54, pp. 64–75, 2017.

[41] Y. O'Connor, W. Rowan, L. Lynch, and C. Heavin, "Privacy by design: informed consent and internet of things for smart health," *Procedia computer science*, vol. 113, pp. 653–658, 2017.

[42] B.-F. Wu, W.-J. Tseng, Y.-S. Chen, S.-J. Yao, and P.-J. Chang, "An intelligent self-checkout system for smart retail," in *2016 International Conference on System Science and Engineering (ICSSE)*. IEEE, 2016, pp. 1–4.

[43] T. Chatzidimitris, D. Gavalas, V. Kasapakis, C. Konstantopoulos, D. Kypriadis, G. Pantziou, and C. Zaroliagis, "A location history-aware recommender system for smart retail environments," *Personal and Ubiquitous Computing*, pp. 1–12, 2020.

[44] B. Heung, H. C. Ho, J. Zhang, A. Knudby, C. E. Bulmer, and M. G. Schmidt, "An overview and comparison of machine-learning techniques for classification purposes in digital soil mapping," *Geoderma*, vol. 265, pp. 62–77, 2016.

[45] E. Symeonaki, K. Arvanitis, and D. Piromalis, "A context-aware middleware cloud approach for integrating precision farming facilities into the iot toward agriculture 4.0," *Applied Sciences*, vol. 10, no. 3, p. 813, 2020.

[46] W. Min, S. Jiang, L. Liu, Y. Rui, and R. Jain, "A survey on food computing," *ACM Computing Surveys (CSUR)*, vol. 52, no. 5, pp. 1–36, 2019.

[47] H. Habibzadeh, A. Boggio-Dandry, Z. Qin, T. Soyata, B. Kantarci, and H. T. Mouftah, "Soft sensing in smart cities: Handling 3vs using

recommender systems, machine intelligence, and data analytics," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 78–86, 2018.

[48] X. Yang, L. Shu, J. Chen, M. A. Ferrag, J. Wu, E. Nurellari, and K. Huang, "A survey on smart agriculture: Development modes, technologies, and security and privacy challenges," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 2, pp. 273–302, 2020.

[49] J. Indulska and P. Sutton, "Location management in pervasive systems," in *Conferences in Research and Practice in Information Technology Series*, vol. 34. Citeseer, 2003, pp. 143–151.

[50] F. Cauteruccio, G. Fortino, A. Guerrieri, A. Liotta, D. C. Mocanu, C. Perra, G. Terracina, and M. T. Vega, "Short-long term anomaly detection in wireless sensor networks based on machine learning and multi-parameterized edit distance," *Information Fusion*, vol. 52, pp. 13–30, 2019.

[51] R. Gravina, P. Alinia, H. Ghasemzadeh, and G. Fortino, "Multi-sensor fusion in body sensor networks: State-of-the-art and research challenges," *Information Fusion*, vol. 35, pp. 68–80, 2017.

[52] J. Zhang, B. Chen, Y. Zhao, X. Cheng, and F. Hu, "Data security and privacy-preserving in edge computing paradigm: Survey and open issues," *IEEE Access*, vol. 6, pp. 18 209–18 237, 2018.

[53] A. Martínez-Ballesté, P. A. Pérez-Martínez, and A. Solanas, "The pursuit of citizens' privacy: a privacy-aware smart city is possible," *IEEE Communications Magazine*, vol. 51, no. 6, pp. 136–141, 2013.

[54] H. Li, J. Wu, Y. Gao, and Y. Shi, "Examining individuals' adoption of healthcare wearable devices: An empirical study from privacy calculus perspective," *International journal of medical informatics*, vol. 88, pp. 8–17, 2016.

[55] Nvidia, "Nvidia jeston." [Online]. Available: https://developer.nvidia.com/buy-jetson

[56] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[57] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Brazilian symposium on artificial intelligence*. Springer, 2004, pp. 286–295.

[58] Y. Carmon, A. Raghunathan, L. Schmidt, P. Liang, and J. C. Duchi, "Unlabeled data improves adversarial robustness," *arXiv preprint arXiv:1905.13736*, 2019.

[59] X. Pu, L. Liu, Y. Mei, S. Sivathanu, Y. Koh, and C. Pu, "Understanding performance interference of i/o workload in virtualized cloud environments," in *2010 IEEE 3rd International Conference on Cloud Computing*. IEEE, 2010, pp. 51–58.

[60] E. Liberty, Z. Karnin, B. Xiang, L. Rouesnel, B. Coskun, R. Nallapati, J. Delgado, A. Sadoughi, Y. Astashonok, P. Das *et al.*, "Elastic machine learning algorithms in amazon sagemaker," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 731–737.

[61] W. Hummer, V. Muthusamy, T. Rausch, P. Dube, K. El Maghraoui, A. Murthi, and P. Oum, "Modelops: Cloud-based lifecycle management for reliable and trusted ai," in *2019 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 2019, pp. 113–120.

[62] D. Baylor, E. Breck, H.-T. Cheng, N. Fiedel, C. Y. Foo, Z. Haque, S. Haykal, M. Ispir, V. Jain, L. Koc *et al.*, "Tfx: A tensorflow-based production-scale machine learning platform," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1387–1395.

[63] M. Zaharia, A. Chen, A. Davidson, A. Ghodsi, S. A. Hong, A. Konwinski, S. Murching, T. Nykodym, P. Ogilvie, M. Parkhe *et al.*, "Accelerating the machine learning lifecycle with mlflow." *IEEE Data Eng. Bull.*, vol. 41, no. 4, pp. 39–45, 2018.

[64] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "Devops," *Ieee Software*, vol. 33, no. 3, pp. 94–100, 2016.

[65] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean, "Characterizing concept drift," *Data Mining and Knowledge Discovery*, vol. 30, no. 4, pp. 964–994, 2016.

[66] E. Bisong, "Kubeflow and kubeflow pipelines," in *Building Machine Learning and Deep Learning Models on Google Cloud Platform*. Springer, 2019, pp. 671–685.

[67] X. He, K. Zhao, and X. Chu, "Automl: A survey of the state-of-the-art," *Knowledge-Based Systems*, vol. 212, p. 106622, 2021.

[68] D. Crankshaw, G.-E. Sela, X. Mo, C. Zumar, I. Stoica, J. Gonzalez, and A. Tumanov, "Inferline: latency-aware provisioning and scaling for prediction serving pipelines," in *Proceedings of the 11th ACM Symposium on Cloud Computing*, 2020, pp. 477–491.

[69] T. Rausch, S. Nastic, and S. Dustdar, "Emma: Distributed qos-aware mqtt middleware for edge computing applications," in *2018 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 2018, pp. 191–197.

[70] E. Jonas, J. Schleier-Smith, V. Sreekanti, C.-C. Tsai, A. Khandelwal, Q. Pu, V. Shankar, J. Carreira, K. Krauth, N. Yadwadkar *et al.*, "Cloud programming simplified: A berkeley view on serverless computing," *arXiv preprint arXiv:1902.03383*, 2019.

[71] Kubernetes, "Kubernetes." [Online]. Available: https://kubernetes.io/

[72] T. Rausch, A. Rashed, and S. Dustdar, "Optimized container scheduling for data-intensive serverless edge computing," *Future Generation Computer Systems*, vol. 114, pp. 259–271, 2021.

[73] R. Want, A. Hopper, V. Falcao, and J. Gibbons, "The active badge location system," *ACM Transactions on Information Systems (TOIS)*, vol. 10, no. 1, pp. 91–102, 1992.

[74] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Blockchain for 5g and beyond networks: A state of the art survey," *Journal of Network and Computer Applications*, p. 102693, 2020.

[75] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Sensing as a service model for smart cities supported by internet of things," *Transactions on emerging telecommunications technologies*, vol. 25, no. 1, pp. 81–93, 2014.

[76] J. Zhang, B. Iannucci, M. Hennessy, K. Gopal, S. Xiao, S. Kumar, D. Pfeffer, B. Aljedia, Y. Ren, M. Griss *et al.*, "Sensor data as a service–a federated platform for mobile data-centric service development and sharing," in *2013 IEEE International Conference on Services Computing*. IEEE, 2013, pp. 446–453.