Towards EDISON: An edge-native approach to distributed interpolation of environmental data

Lauri Lovén, Ella Peltonen, Abhinay Pandya, Teemu Leppänen, Ekaterina Gilman, Susanna Pirttikangas, Jukka Riekki
Center for Ubiquitous Computing, University of Oulu, Oulu, Finland
Email: first.last@oulu.fi

Abstract—**Prevalent weather prediction methods are based on sensor data, collected by satellites and a sparse grid of stationary weather stations. Various initiatives improve the prediction models by including additional data sources such as mobile weather sensors, mobile phones, and micro weather stations of, for example, smart homes. The underlying computing paradigm is predominantly centralized, with all data collected and analyzed in the cloud. This solution is not scalable. When the spatial and temporal density of weather sensor data grows, the required data transmission capacities and computational resources become unfeasible. We identify the challenges posed by spatial distribution of a weather prediction model, and suggest solutions for those challenges. We propose EDISON: an edge-native interpolation approach based on AI methods, distributed horizontally on edge servers. Finally, we demonstrate EDISON with a simple, simulated setup.**

I. Introduction

More than half of the world's population lives in cities. By 2050, this number is predicted to increase to 70 percent[1]. Development of information and communication technologies plays an essential role in transforming cities into smart cities, aiming to improve city efficiency and sustainability. City-scale sensing technologies and data-driven solutions provide the building blocks for novel smart applications, enlarge business opportunities, and improve the development of urban services across domains and stakeholders. Cities are already full of sensor-equipped technologies [1], such as water and electric meters, and sensors measuring traffic, buildings and weather. Patterns, anomalies and events identified in the data provide novel insights and help preparing for unforeseen scenarios, and in city planning.

However, several challenges need to be tackled before these benefits can be fully realized. The massive scale and heterogeneity of city data requires modelling and analytics to understand city activity. Security and privacy must to be guaranteed as well. These solutions need to offer a feasible trade-off between cost and quality to justify the investment. Moreover, smart cities rely on various data sources, such as sensing devices, spatial data, user contributed content, and data available from authorities and services. Sensor data is produced by a mixed set of heterogeneous sensors and computation nodes with varying resources and capabilities. For example, high-end sensors produce high-quality data and are intelligent and capable of, e.g., self-calibration or performing some data analytics autonomously, whereas cheap sensors produce lower quality sensor data that requires calibration and subsequent data analytics. Furthermore, sensor connectivity may be intermittent and at times low in bandwidth, which makes cloud-based solutions infeasible.

To address these challenges, we propose a distributed, decentralized approach, based on edge computing and artificial intelligence (AI), for environmental sensing and sensor data modeling at a city scale. We focus on weather sensing and modelling, paving way for other large-scale sensing applications in mobile city environments using edge computing capabilities. For example, smart city services like traffic management and emergency response rely on weather prediction models [1]. Yet building such models without accurate ground truth for calibration and training is challenging. Moreover, the models need to be adaptable to changes in the state of the city.

In this work, we look at large-scale environmental sensor networks and the models used to analyze their data. In particular, we concentrate on interpolation models which extend the observations of a sparse sensor network to those areas and points in time where no observations are available. Section II looks into the state of the art on the subject, and section III details the challenges and related research questions. In section IV we outline a novel, edge-native, AI-based interpolation architecture, EDISON, and detail the steps necessary for implementing that architecture, while section V presents a simulated example with a preliminary prototype model based on EDISON.

II. Related work

City-scale computing: Some platforms are suggested for city scale computation activities [2], [3]. However, existing solutions require data aggregation and processing in the cloud, imposing several challenges such as high latencies, high transmission costs, and loss of privacy [4]. For example, when real-time decision-making is required, such as with autonomous vehicles, high latencies for centralized sensor data collection and real-time feedback are untenable.

Further, Internet of Things (IoT) devices producing data are often limited in computational and transmission resources [5].While instantaneous cloud-based operation is not practical, bringing computations closer to the participating devices in the edge computing model tackles the cloud challenges [6], [7].

[1]United Nations, https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html

Edge computing for smart cities: A number of smart city studies apply the edge computing paradigm. For instance, Hossain et al. [8] present an edge computing framework for situation awareness in an IoT-based smart city. First experiments were conducted in terms of latency and situation awareness when raw IoT data is processed at the edge devices. Cicirelli et al. [9] present an agent-based, distributed platform for managing a network of computing nodes, spread within a city. Computation is executed at the edge as well as the cloud, which handles computationally demanding tasks.

City-scale sensing and data analytics: Smart cities rely on IoT, big data, cyber-physical systems, and cloud computing technologies [2], [10]. Analysing the huge amount of data available is challenging and time-consuming. Smart city applications use different processing approaches, e.g., batch and stream processing, supported by various big data architectures. Such solutions, however, are cloud-based, and hence have their challenges. Some studies do concentrate on edge-based platforms, but they are not focusing on distribution of interpolation and analytics models [11].

Intelligent transportation systems: Efficiency, safety and ecology are essential properties for intelligent transport systems (ITS) in smart cities. City level digitalisation for situation awareness [12], [13], [14] is well on the way around the world[2] and data from different phenomena in the cities is starting to be available. Availability of data enables AI-based prediction of transport flows, congestion, and need for maintenance, generally planning for a more sustainable, pleasant and prosperous city life. In essence, ITS aims to create decision support in the special case of mobility. Information on weather and driving conditions is essential for traffic safety [15], [16]. Road weather information systems (RWIS) rely, e.g., on road weather stations (RWSs), observing road weather and the driving conditions. However, RWSs are sparse as professional weather stations (including sensor, electronics, mast, and power supply) are expensive. Building RWISs with higher spatial densities requires serious investment.

III. Challenges

The challenges for city scale sensor data interpolation can be separated to resource, transmission, data, and distribution layers. Some particular challenges for each layer are listed in the subsections below; Table I details the related research questions on each layer.

1) Resource: If sensors are battery-powered, they may stop sending data due to the energy saving mode or an empty battery. Further, some sensors have less bias and variance in their observations than others. For example, expensive weather sensors deployed at dedicated weather stations produce better observations than cheap thermometers. Properly calibrated, the cheap thermometers can still contribute to an understanding about local variance. However, large-scale sensor calibration is challenging. Further, sensors may experience *drift*, i.e., their observations may slowly gain bias or variance. Drift calls for online or continuous calibration, posing further challenges.

[2]https://arrayofthings.github.io/, https://www.hudsonyardsnewyork.com/, https://www.c40.org/, accessed 12th April 2019.

TABLE I
Central research questions of large-scale environmental data interpolation.

Layer	Research questions
Resource	How does the interpolation model handle data missing due to energy constraints? Are sensors self-calibrating, or is some external entity managing the calibration? How much computational resources can be used during and after the calibration? How is data made accessible for the calibration? How to implement on-line calibration? How to mitigate for faulty installations? How to mitigate for crowd-sourcing errors?
Transmission	How does the interpolation model handle data missing due to intermittent connectivity? What parts of the interpolation model can or should be computed locally and what offloaded? If a location was observed recently, how long will that observation be valid? How does the uncertainty related to an observation change after the observation? How and where to store the data produced by mobile sensors?
Data	How to handle big data? How does the interpolation model handle data missing due to the spatial or temporal scarcity of sensors? How does the distribution take into correlation both short and long term covariance structures in the data? How to handle drift?
Distribution	How can local models exchange information effectively? How to implement scalable and reliable orchestration for local model data exchange? How to reach consensus among decentralized, co-operative local models.

Additional challenges are caused by the misplacement of the sensing devices, e.g., near exhaust pipes or heat sinks, or improper data collection procedures, e.g. in crowd-sourcing [17]. Relying only on local data processing and analysis may cause errors resulting from faulty measurements.

2) Transmission: Sensors may be mobile, in which case they may occasionally observe locations that no stationary sensors see. However, mobility makes distributed data management difficult especially for rapidly moving sensors such as those on vehicles and trains. Further, mobile sensor connectivity may be intermittent, leading to data loss.

3) Data: Sensors may be deployed in a geographically small or wide area, densely or sparsely. The wider the area, the more spatial variability needs to be considered. The denser the network, the more data is produced, calling for Big Data methodologies or distributed modelling [18]. Varying spatial densities, due to for example the popularity of some mobile sensor trajectories, may produce biased models if not accounted for. Further, sensors may produce observations frequently or less so. Varying frequencies will introduce sparsity and synchronization challenges. Finally, environmental – in particular weather-related – data typically incorporates complex short and long term covariance structures, both spatially and temporally. An interpolation model not considering these covariance structures risks providing faulty interpolations.

4) Distribution: Large setups may require the distribution of data as well as the computations related to interpolation model training and inference. Since data is produced locally, data is naturally distributed based on sensor locations. Local data, however, does not include the possible long-distance covariance structures present in the global data. Models based on local data thus need a way of exchanging information. Information exchange can be implemented either centrally, by an orchestrator, or cooperatively among the decentralized local data managers. Orchestration introduces scalability and reliability challenges while co-operative model building requires potentially complex consensus algorithms [19], [20].

There are several approaches to distributed model building. For example, ensemble learning methods create a set of models to be used together, while federated learning relies on the collaborative learning of a shared model, using training data subsets available on the devices [21]. Such approaches provide a number of benefits, like reducing latency and supporting privacy, as data stays at the node and is not shared with the peers [4]. However, while some propositions are starting to appear [22], the question of decentralized model building without cloud coordination is still an open question.

IV. EDISON

We propose EDISON, an edge-native, AI-based distributed approach to interpolating the observations of a sensor network. The sensor network can be heterogeneous, with a sparse set of mobile and stationary sensors, observing the environment with varying temporal frequencies in a wide spatial area. The interpolation model provides predictions of the variables of interest on a dense spatio-temporal grid, based on the sparse observations. As such, the model captures the spatio-temporal dependency structure of the variables in the observed area, and use that structure to interpolate between sparse observations.

EDISON comprises data collection, distribution, local model building, and model calibration. Data is collected by a sensor network with edge connectivity. Distribution partitions the data for the local interpolation models, which predict the values of variables in places with no observations. Finally, model calibration shares information between the local models, ensuring they are in agreement with each other. Each part of the method is further detailed below.

1) Data collection: To ensure some data is always available for interpolation, EDISON expects some of the sensors in the network to be stationary, with fixed connections. However, the majority of the sensors can be mobile. EDISON assumes the sensors are connected to edge servers with configurable networks, i.e., such that the edge server serving each sensor can be configured (by, e.g., SDN) at least during system setup. This requirement stems from EDISON grouping the sensors based on the spatial density of data. Fig. 1 illustrates an urban sensor network setup with high-quality stationary sensors, and a number of mobile sensors installed on, e.g., vehicles or drones, with varying fidelity.

2) Distribution: EDISON distributes the data as well as model computations on edge servers. First, EDISON partitions the spatial area into a dense grid of rectangular cells. The cells are all grouped into small connected areas, referred to as *observation regions*, with a similar distribution of historical observations on each region (Fig. 2). This reduces bias for those areas which are densely observed.

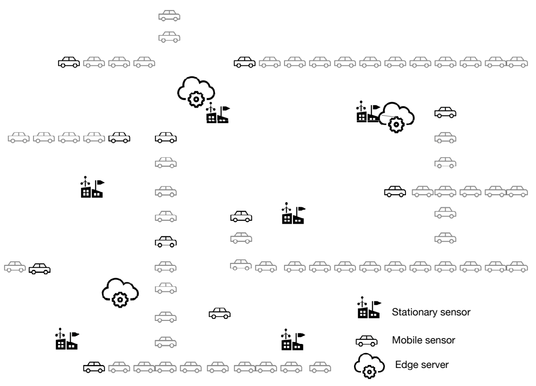

Fig. 1. Example of EDISON sensor setup. The setup comprises stationary sensors (e.g., RWSs), mobile sensors (e.g., installed in vehicles), and edge servers. Mobile sensor traces are depicted in gray.

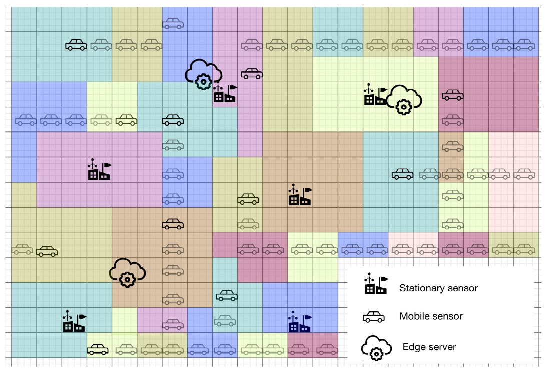

Fig. 2. Example grouping of spatial cells (squares in the underlying grid) into connected observation regions (colored areas). The resulting observation regions have roughly a similar distribution of observations.

The observation regions are further allocated to *edge locales*, with edge servers ideally positioned near the center of each edge locale. To allow for model calibration (see Section IV-4 below), the edge locales overlap such that each observation region is allocated to two edge locales, both centred by nearby edge servers. Fig. 3 illustrates three overlapping edge locales, centred by three edge servers.

3) Local models: Edge servers are responsible for creating and managing local interpolation models for their edge locales. Each model estimates the value of variables in observation regions where the sparse sensor network has no recent observations. The local models consider short and long term temporal covariance structures as well as short distance (within edge locales) spatial covariance structures between and among the observations of each variable.

4) Model calibration: To account for long distance (between edge locales) spatial covariance structures in the data, the local models exchange information. In more detail, the

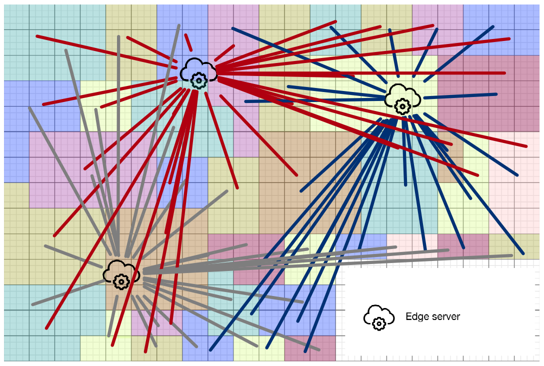

Fig. 3. Example of three overlapping edge locales, each centred by an edge server. The lines denote the observation regions belonging to each of the three edge locales.

parameters of each local model are updated to mitigate the differences between the interpolations of the neighbouring edge clusters with overlapping observation regions. If a cloud connection is available, EDISON can use a centralized orchestrator for forcing a consensus between the local models. Lacking cloud access, EDISON edge servers negotiate the consensus among themselves.

Algorithm 1 outlines our approach. Each edge server builds a local interpolation model. Periodically (after receiving new observation data of some quantum), each edge server updates its own model and communicates this new model to the neighboring edge servers with overlapping observation regions. When an edge server receives a model update from a neighboring edge server, it evaluates the accuracy of the new model against its own local model. If the accuracy of the received model is higher than its own, the edge server performs model calibration to achieve consensus. The decision to re-calibrate the model also takes into account the computational effort in re-training as against the accuracy difference between the models. This process will go on ad infinitum to ensure the local models are current and consistent.

V. Simulated Example

We constructed a simple example of EDISON processing spatial data. Some of the steps outlined in Section IV were omitted for simplicity and brevity.

A. Data

The example sensor network is deployed in a rectangular area, consisting of 101 $\times$ 101 rectangular cells. The network has 440 stationary, high-quality sensors spread evenly (at coordinates divisible by 5) across the area. Mobile and stationary sensors have produced an equal number of observations in every cell, so that observation regions are equal to the cells. 61 edge servers (5 $\times$ 5 "blue" and 6 $\times$ 6 "red") and their respective edge locales are placed such that each cell is covered by two edge locales. Figure 4 illustrates the stationary sensors, the edge servers, and the edge locales.

120 spatial data frames, comprising the sensor data for each cell (each frame thus consisting of 101 $\times$ 101 observations),

Algorithm 1: Edge-native distributed interpolation

```
Input:
The $n \times m$ rectangular spatial grid $G$.
Rectangular spatial cells $g_{i,j} \in G$.
$M$ edge servers.
Output:
A set of local interpolation models, one for each edge server,
which are in consensus with the models of their neighboring
edge servers.
The interpolations inferred by local models.
// Set up the observation regions and the
   edge locales.
1  Assign each $g_{i,j}$ to connected $ObservationRegions$ $O_{1...k}$
     such that the distribution of historical observations in each $O_i$
     is approximately equal.
2  Assign each $O_i$ to two connected $EdgeLocales$ $E_{1...M}$,
     $O_i \in E_x$ and $O_i \in E_y$. Minimize the distance between $O_i$
     and the centers of $E_x$, $E_y$.
3  Allocate one edge server for each edge locale. Minimize the
     distance between edge servers and the centers of the edge
     locales.
// The following procedure runs at each
   edge server.
4  while True do
5      if timer $\geq$ threshold AND enough new observations have
         arrived then
6          Update the local model.
7          Send the local model (sufficient statistics) to all edge
             servers covering the observation regions in the edge
             locale.
8      if $ModelUpdateFromNeighboringEdgeServer$ then
9          Evaluate the received model update for the recent data.
10         if (accuracy($LocalModel$) $\geq$
             accuracy($ReceivedModel$) then
11             Send the local model (sufficient statistics) to all
                 edge servers covering the observation regions in
                 the edge locale.
12         else
13             Perform model calibration.
14             Send the new model (sufficient statistics) to all
                 edge servers covering the observation regions in
                 the edge locale.
```

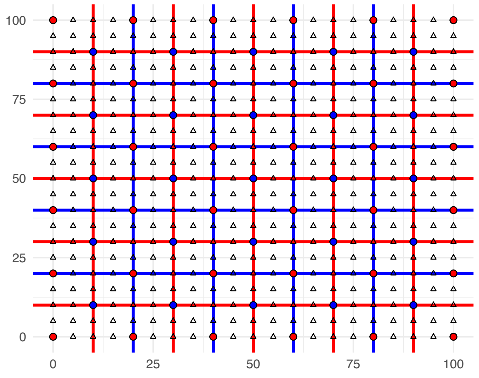

Fig. 4. Edge servers (red and blue dots), centring the rectangular edge locales they serve (bordered by the respective nearest red or blue solid lines). Stationary sensors shown as black triangles.

were simulated with a combination of Gaussian processes with short and long term covariance functions. 100 of the frames were labeled as the training set, for training the interpolation models, while 20 frames were labeled as the test set and set aside for testing model quality. The simulation procedure is further detailed below:

1) A grid with a short distance spatial covariance structure, representing a contour map, was sampled from a Gaussian process with exponential covariance function.
2) A grid with a long distance spatial covariance structure was sampled from a Gaussian process with Matern covariance function.
3) Short and long distance covariance grids were summed item-wise.
4) 120 frames, each a grid with long-distance covariance structure (representing, e.g., temperature) were sampled from a Gaussian process with exponential covariance function.
5) The grid resulting from step 3 was item-wise added to each grid from step 4, resulting in 120 data frames.

The process is illustrated in Fig. 5. Data was generated with the R gstat package [23], [24], parameters for each phase can be found in Table II.

TABLE II
SAMPLING PARAMETERS FOR EACH PHASE OF DATA SIMULATION.

Step	Cov. funct.	range	κ	anisotropy	$(\beta_0, \beta_x, \beta_y)$	N
1	Exponential	40	1	(45, 0.5)	(0, 0.3, 0)	1
2	Matern	80	1		(0, 0, 0)	1
4	Matern	80	1	(90, 0.9)	(15, 0, 0)	120

B. Models

We used the Kriging method (with R's gstat package [23], [24]) for interpolating data between the stationary sensors. We trained a local interpolation model (i.e., fitted a variogram) for each edge locale, using only the data produced within the locale, averaged over the 100 training frames. We used the local models for interpolating data between the stationary sensors in each edge locale, in each of the 20 frames in the test set. After interpolation, we simulated the first round of consensus among the local models by averaging the overlapping predictions for each cell. For comparison, we also simulate a global model, with access to full training data comprising the whole grid.

C. Results

Root mean square errors (RMSE) between the ground truth and the interpolated values by the global model, the local models, and the first round of consensus between the local models can be found in Table III. While the global model is slightly ahead, the RMSEs of the local models are indeed improved by the model averaging, simulating consensus. Fig. 6 illustrates the ground truth for the first testing frame as well as the interpolations of the global model, the local models, and the consensus.

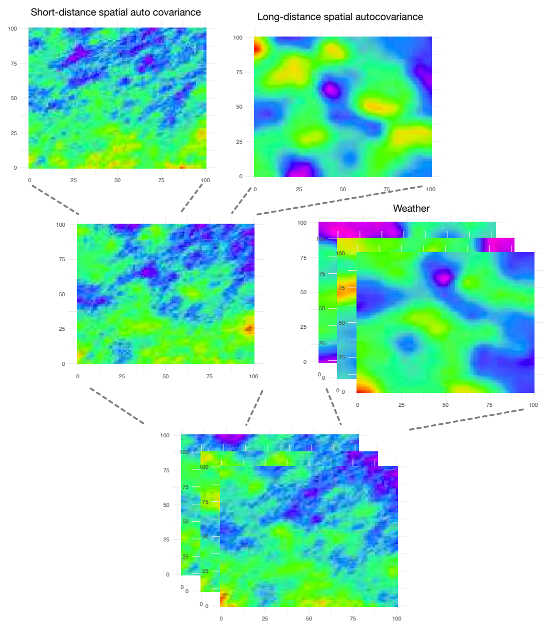
Short-distance spatial auto covariance
Long-distance spatial autocovariance
Weather

Fig. 5. Composition of simulation data. A contour map, representing short distance covariance structures, is added to a long distance spatial covariance map. The result is added to 120 temperature maps, producing finally 120 spatial data frames consisting of multiple types of spatial covariance.

VI. Conclusion and Discussion

Providing fine-grained reliable weather predictions requires large amounts of data and processing resources. However,

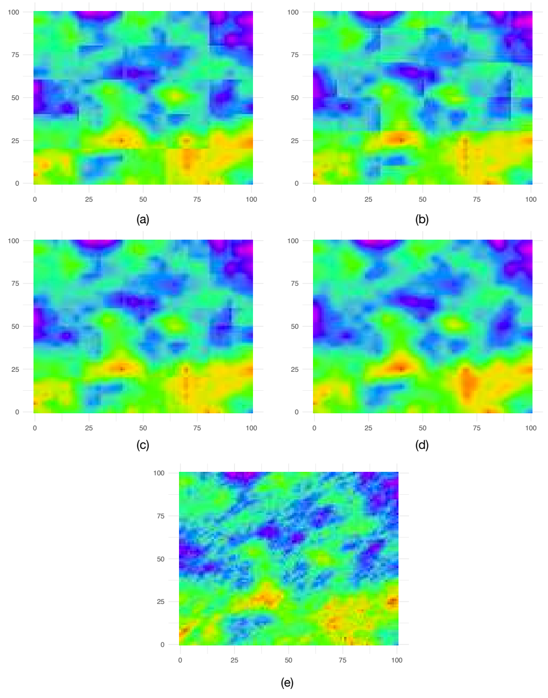

Fig. 6. Interpolation results on the first testing frame. Included are the interpolations with the local models for blue (a) and red (b) edge locales, their average (c), simulating consensus, the global model (d) and the ground truth (e).

TABLE III
AVERAGED RMSEs OVER THE TEST SET OF THE SIMULATED EXAMPLE.

Model	**RMSE**
Blue local models	0.61
Red local models	0.63
Consensus	0.59
Global interpolation	0.57

often such computational resources are not available for particular scenarios and interpolation is required. This article outlined EDISON, an edge-native distributed AI method for interpolating the observations of a heterogeneous and sparse set of mobile and stationary sensors. By partitioning the spatial area to overlapping locales, EDISON develops local prediction models with calibration capabilities. This way, EDISON is able to achieve the full advantages of distributed processing, as well as enjoy the possibilities of model tuning to minimize either local measurement errors or adhere to global changes not captured locally.

This study included a simple simulated example, which demonstrated EDISON functionality. While the example only simulated the first round of consensus among the local models, it resulted in an improvement in interpolation quality.

For future work, we are interested in further elaborating the consensus mechanism and its convergence, and testing EDISON with real data. We will also experiment with different interpolation and calibration models for EDISON. Finally, we will consider whether agent-based solutions provide benefits for EDISON.

ACKNOWLEDGMENT

This research is supported by the Academy of Finland 6Genesis Flagship (grant 318927) program, the MEC-AI project, funded by the Future Makers program of Jane and Aatos Erkko Foundation and Technology Industries of Finland Centennial Foundation, by the EU Horizon 2020 project CUTLER: Coastal Urban developmenT through the LEnses of Resiliency, under contract no. 770469 (http://www.cutler-h2020.eu/), and the personal grant for Lauri Lovén on Edge-native AI research by the Tauno Tönning foundation.

REFERENCES

[1] M. Naphade, G. Banavar, C. Harrison, J. Paraszczak, and R. Morris, “Smarter cities and their innovation challenges,” *Computer*, vol. 44, no. 6, pp. 32–39, 2011.

[2] E. F. Z. Santana, A. P. Chaves, M. A. Gerosa, F. Kon, and D. Milojicic, “Software platforms for smart cities: Concepts, requirements, challenges, and a unified reference architecture,” vol. 50, no. 6, 2016. [Online]. Available: http://arxiv.org/abs/1609.08089

[3] H. Mehmood, E. Gilman, and M. Cortes, “Implementing big data lake for heterogeneous data sources,” in *Proc. 1st Int. Workshop onData-Driven Smart Cities, in conjunction with 35th IEEE Int. Conf. on Data Engineering (ICDE 2019)*, Macau SAR, 0.

[4] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, “Wireless Network Intelligence at the Edge,” *arXiv preprint*, 2018.

[5] U. Raza, A. Camerra, A. L. Murphy, T. Palpanas, and G. P. Picco, “What does model-driven data acquisition really achieve in wireless sensor networks?” in *2012 IEEE International Conference on Pervasive Computing and Communications, PerCom 2012*, no. (19-23 March 2012). Lugano, Switzerland: IEEE, 2012, pp. 85–94.

[6] L. Lovén, T. Leppänen, E. Peltonen, J. Partala, E. Harjula, P. Porambage, M. Ylianttila, and J. Riekki, “EdgeAI: A vision for distributed, edge-native artificial intelligence in future 6G networks,” in *The 1st 6G Wireless Summit*, Levi, Finland, 2019, pp. 1–2.

[7] E. Peltonen, T. Leppänen, and L. Lovén, “EdgeAI: Edge-native distributed platform for artificial intelligence,” in *The 1st 6G Wireless Summit*, Levi, Finland, 2019, pp. 1–2.

[8] S. K. A. Hossain, M. A. Rahman, and M. A. Hossain, “Edge computing framework for enabling situation awareness in IoT based smart city,” *Journal of Parallel and Distributed Computing*, vol. 122, pp. 226–237, 2018.

[9] F. Cicirelli, A. Guerrieri, G. Spezzano, and A. Vinci, “An edge-based platform for dynamic Smart City applications,” *Future Generation Computer Systems*, vol. 76, pp. 106–118, 2017.

[10] G. Fortino, W. Russo, C. Savaglio, M. Viroli, and M. Zhou, “Modeling opportunistic IoT services in open IoT ecosystems,” *WOA*, pp. 90–95, 2017.

[11] E. Lagerspetz, S. Varjonen, F. Concas, J. Mineraud, and S. Tarkoma, “Demo: MegaSense: Megacity-scale accurate air quality sensing with the edge,” *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom ’18)*, pp. 843–845, 2018.

[12] M. M. Rathore, A. Paul, W. H. Hong, H. C. Seo, I. Awan, and S. Saeed, “Exploiting IoT and big data analytics: Defining smart digital city using real-time urban data,” *Sustainable Cities and Society*, vol. 40, no. November 2017, pp. 600–610, 2018. [Online]. Available: https://doi.org/10.1016/j.scs.2017.12.022

[13] K. R. Malik, Y. Sam, M. Hussain, and A. Abuarqoub, “A methodology for real-time data sustainability in smart city: Towards inferencing and analytics for big-data,” *Sustainable Cities and Society*, vol. 39, no. November 2017, pp. 548–556, 2018.

[14] S. Pirttikangas, E. Gilman, X. Su, T. Leppanen, A. Keskinarkaus, M. Rautiainen, M. Pyykkonen, and J. Riekki, “Experiences with smart city traffic pilot,” *Proceedings - 2016 IEEE International Conference on Big Data, Big Data 2016*, pp. 1346–1352, 2016.

[15] L. Lovén, V. Karsisto, H. Järvinen, M. J. Sillanpää, T. Leppänen, E. Peltonen, S. Pirttikangas, and J. Riekki, “Mobile road weather sensor calibration by sensor fusion and linear mixed models,” *PLOS ONE*, vol. 14, no. 02, pp. 1–17, 2019.

[16] V. Karsisto and L. Lovén, “Verification of road surface temperature forecasts assimilating data from mobile sensors.” *Weather and Forecasting*, vol. 34, pp. 539–558, 2019.

[17] M. Budde, A. Schankin, J. Hoffmann, M. Danz, T. Riedel, and M. Beigl, “Participatory sensing or participatory nonsense? – Mitigating the effect of human error on data quality in citizen science,” *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 1, no. 3, pp. 39:1–39:23, 2017.

[18] M. Chen, S. Mao, and Y. Liu, “Big data: A survey,” *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171–209, 2014.

[19] S. Li, H. Du, and X. Lin, “Finite-time consensus algorithm for multi-agent systems with double-integrator dynamics,” *Automatica*, vol. 47, no. 8, pp. 1706–1712, 2011.

[20] R. Lazzeretti, S. Horn, P. Braca, and P. Willett, “Secure multi-party consensus gossip algorithms,” in *Proc. 2014 IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP)*, 2014, pp. 7406–7410.

[21] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan, T. Van Overveldt, D. Petrou, D. Ramage, and J. Roselander, “Towards federated learning at scale: system design,” in *The 2nd SysML Conference*, Palo Alto, CA, USA, 2019.

[22] H. Kim, J. Park, M. Bennis, and S.-L. Kim, “On-Device federated learning via blockchain and its latency analysis,” *arXiv preprint*, pp. 1–4, 2018.

[23] E. J. Pebesma, “Multivariable geostatistics in S: The gstat package,” *Computers and Geosciences*, vol. 30, no. 7, pp. 683–691, 2004.

[24] B. Gräler, E. Pebesma, and G. Heuvelink, “Spatio-Temporal Interpolation using gstat,” *The R Journal*, vol. 8, no. 1, pp. 204–218, 2016.