sensors

MDPI

Article

EDISON: An Edge-Native Method and Architecture for Distributed Interpolation

Lauri Lovén [1,*], Tero Lähderanta [2], Leena Ruha [2,3], Ella Peltonen [1], Ilkka Launonen [2], Mikko J. Sillanpää [2], Jukka Riekki [1] and Susanna Pirttikangas [1]

[1] Center for Ubiquitous Computing, University of Oulu, FI-90014 Oulu, Finland; ella.peltonen@oulu.fi (E.P); jukka.riekki@oulu.fi (J.R.); susanna.pirttikangas@oulu.fi (S.P.)
[2] Research Unit of Mathematical Sciences, University of Oulu, FI-90014 Oulu, Finland; tero.lahderanta@oulu.fi (T.L.); leena.ruha@luke.fi (L.R.); ilkka.launonen@oulu.fi (I.L.); mikko.sillanpaa@oulu.fi (M.J.S.)
[3] Natural Resources Institute Finland, FI-90014 Oulu, Finland
* Correspondence: lauri.loven@oulu.fi

Abstract: Spatio-temporal interpolation provides estimates of observations in unobserved locations and time slots. In smart cities, interpolation helps to provide a fine-grained contextual and situational understanding of the urban environment, in terms of both short-term (e.g., weather, air quality, traffic) or long term (e.g., crime, demographics) spatio-temporal phenomena. Various initiatives improve spatio-temporal interpolation results by including additional data sources such as vehicle-fitted sensors, mobile phones, or micro weather stations of, for example, smart homes. However, the underlying computing paradigm in such initiatives is predominantly centralized, with all data collected and analyzed in the cloud. This solution is not scalable, as when the spatial and temporal density of sensor data grows, the required transmission bandwidth and computational capacity become unfeasible. To address the scaling problem, we propose EDISON: algorithms for distributed learning and inference, and an edge-native architecture for distributing spatio-temporal interpolation models, their computations, and the observed data vertically and horizontally between device, edge and cloud layers. We demonstrate EDISON functionality in a controlled, simulated spatio-temporal setup with 1 M artificial data points. While the main motivation of EDISON is the distribution of the heavy computations, the results show that EDISON also provides an improvement over alternative approaches, reaching at best a 10% smaller RMSE than a global interpolation and 6% smaller RMSE than a baseline distributed approach.

Keywords: edgeAI; edge computing; interpolation; distributed AI; distributed computing; kriging

check for updates

Citation: Lovén, L.; Lähderanta, T.; Ruha, L.; Peltonen, E.; Launonen, I.; Sillanpää, M.J.; Riekki, J.; Pirttikangas, S. EDISON: An Edge-Native Method and Architecture for Distributed Interpolation. *Sensors* **2021**, *21*, 2279. https://doi.org/10.3390/s21072279

Academic Editor: Marco Picone

Received: 28 February 2021
Accepted: 22 March 2021
Published: 24 March 2021

1. Introduction

More than half of the world's population lives in cities and by 2050, this number is predicted to increase to nearly 70% [1]. Increased numbers of people populating ever smaller areas of land increases the need for development of information and communication technologies, to support access and reliability of networking services in the urban environment. This societal and technological development plays an essential role in the development of smart cities, aiming to improve efficiency, sustainability and resilience of both the city itself but also urban networking infrastructure [2].

City-scale sensing technologies and data-driven solutions can also be seen as an enabler for novel smart applications [3,4]. These applications and services can support e.g., sustainability of the building blocks, enlarge business opportunities, and improve the development of urban services across domains and stakeholders. Keeping up and further improving sustainability [5] has already affected many of the aspiring smart cities, which are full of sensor-equipped technologies [2,6], such as water and electric meters, and sensors measuring traffic and weather. Patterns, anomalies and events identified in the data provide novel insights and help to prepare for unforeseen scenarios in city planning. However, several challenges related to the networking architecture itself need to be tackled before these benefits can be fully realized.

From a data point of view, the ever evolving massive scale of the city-driven data [3] requires novel data preprocessing and management technologies on a scale not seen before. The data sources include, e.g., various sensing devices, smart traffic and vehicles, spatial data, user-contributed content, and data available from authorities, businesses, private citizens, and various different services [7]. The heterogeneity of sensors and the wildly varying urban data sources require advanced modeling and analytics technologies suitable for understanding city activities. Because there are people both generating the data and using it [6], security and privacy must be guaranteed as well. Whatever the solutions to tackle these challenges, they need to offer a feasible trade-off between cost and quality to justify the investment, especially for municipalities which are the master operator for different urban computing activities and platforms [8].

Moreover, computational and networking capabilities need to match the increasing needs of services and applications available. City-scale sensor networks and data alone do not usually provide the computational capabilities for further intelligent operations, especially if the main target is to understand the whole contextual and situational picture of the urban environment. In addition, connectivity of the data providers, especially moving objects such as vehicles and carry-on smart devices, may be intermittent and at times low in bandwidth [9]. At the same time, understanding the situational and contextual picture of what is happening in the city requires, in some use cases, real-time data processing [10] in terms of milliseconds to seconds to adjust for e.g., use of emergency services and smart traffic operations.

These challenges of (1) large-scale data, (2) heterogeneous data providers, (3) mobile and low-capability devices as an integral part of the system, and (4) real-time requirements of many urban services, can make traditional cloud-based solutions infeasible. The edge computing paradigm is nowadays suggested to become a key driver to solve these urban computing challenges [11–15], expanding from a single site to smart city scale [16,17]. Application request can be generated at a distance, e.g., in a cloud, but the actual data processing occurs at the edge. Particular benefits are seen in (1) in-network processing of massive-scale heterogeneous data from different domain sources, deemed unrealistic for cloud platforms [18], (2) the low latency provided by edge [12,19], crucial for smart safety, emergency, and health scenarios, and (3) location-awareness and edge computing architectures simplifying, respectively, city network structure and information flow [9,19].

However, no edge architecture is yet ready to become city-scale from the current "IoT-scale", managing the operations of a single home, building, or factory. Even with the latest edge computing platforms, a number of challenges must be solved. These are especially the proper data analytics capabilities and results delivery, currently discussed under the topic of intelligent edge or EdgeAI [20]. In this development, not only running distributed machine leaning or artificial intelligence algorithms on the edge platform is important [21,22], but also collecting, storing, pre-processing, integrating, and fusing the heterogeneous data from various urban sources. Further, large-scale urban areas set physical geographical challenges [11] with various participating devices from stationary buildings to moving vehicles, buses, taxes, and driving-assisted or self-driving cars.

These challenges we have outlined are, of course, impossible to solve in a single scientific article. In our previous work, we have proposed a distributed architectural approach based on the EdgeAI paradigm [20] and tested these preliminary EdgeAI methods with road-weather forecasting using distributed sensor fusion and linear mixed models [23]. With our interest in physical geographical areas, we now study environmental sensing, sensor data modeling, and spatio-temporal interpolation of the data to unobserved regions at a city scale. We focus especially on two of the challenges named above, namely, large-scale data produced by mobile and low-capability devices, briefly touching a potential approach for real-time support in the Discussion section (Section 5). Indeed, Gaussian

Process regression, a popular method for non-linear regression and interpolation of spatially and temporally irregular observations, has computational complexity relative to the N^3, where N is the number of observations [24]. It is clear that such a method is untenable in the data-rich environments of smart cities.

We have provided an early vision for combining multiple data sources to a city-scale computing environment by using edge computing capabilities, an architecture and related analysis methods we call EDISON [25]. The early vision described some of the challenges as well as their possible solutions in edge-native spatio-temporal interpolation, and outlined a potential approach, but lacked a description of the methods and algorithms required, a rigorous evaluation, and an edge-native architecture with communication links between the devices. In this paper, we elaborate the EDISON approach, describing the architecture as well as the methods and algorithms in detail, and provide an extensive evaluation in a simulated and controlled environment. Further, contrary to our other previous work that utilized real-life data to showcase the feasibility and applicability of the calibration method employed by EDISON [23], we now focus on evaluating EDISON distributed learning and inference methods with controlled, simulated data to present a viable alternative in the area of city-scale data-driven edge computing.

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 2 first looks into the state of the art on the subject. In Section 3 we outline EDISON, a novel, edge-native interpolation architecture, and detail the related methods, while Section 4 presents a simulated example with a preliminary prototype model based on EDISON. Finally, in Section 5 we discuss the results, while Section 6 concludes the study. The contributions of this paper can be summarised as the following:

1. We present an edge-native, distributed interpolation architecture for the smart city networking environment, characterized by spatio-temporal nature and large-scale communications.
2. We present a distributed learning and inference method for our architecture, to make edge-native interpolations with spatio-temporally distributed data.
3. We evaluate our solution with a controlled environment of simulations, enforcing the natural phenomena observed in our previous work [23].

2. Related Work

Edge computing. The terminology and the definitions of concepts in edge computing are not fully agreed upon. For example, proponents of the fog computing model consider fog to be a continuum of computing resources along the path from devices to the cloud, and identify edge computing with the devices and their users [26]. On the other hand, edge computing proponents consider edge to comprise resources for communication, computation, control and storage, in close proximity to the devices and end-users, with those resources ranging from light devices to small-scale edge data centers [15]. In this article, we follow the terminology of the edge computing proponents, and consider a three-layer model with a remote cloud, local edge computing servers, and finally the devices.

City-scale computing. The highlighted challenges of city-scale computing can coarsely be summarised as (1) large-scale data quantity and how to process it efficiently [3], (2) heterogeneous data providers in terms of data quality, source (varying from private citizens to vehicles and industrial applications), and sampling frequency [6,7], (3) mobile and low-capacity devices as a part of the system, especially private carry-on devices and vehicles [10,27], and (4) real-time requirements to produce efficient recommendations, situational awareness, and other big-picture services and applications [10]. Some platforms are suggested for city-scale computation activities, including both "traditional" elastic cloud services [28] and data lakes [29].

Further, Internet of Things (IoT) devices, sensors, and different carry-on devices producing data are often limited in computational and transmission capabilities [30]. The current, heavily cloud-based solutions require data aggregation and processing in a remote computational environment, imposing several challenges such as high networking load and latency, high transmission costs, and loss of privacy [16,21,22]. For example, when real-time decision-making is required, such as with autonomous vehicles, high latency for centralized sensor data collection and real-time feedback are untenable. Thus, instantaneous cloud-based operation seems not to be practical at least with real-time contextual data. It is suggested that bringing computations closer to the participating devices in the edge computing model tackles the cloud challenges [20,31].

Edge computing for smart cities. Today, the current research trend agrees that whenever cloud-only architectures are not feasible anymore, edge computing paradigm needs to emerge into the city-scale environment. For instance, Hossain et al. [32] present an edge computing framework for situation awareness in an IoT-based smart city. Their first experiments consider latency and situation awareness when raw IoT data is processed at the edge devices, with a multi-layer architecture. However, they utilize the edge only for data processing, demanding the cloud services for the final combination of the data and running learning models. This is, by our understanding, not meeting the real-time requirements when not only processing but also the delivery of results should be considered in a timely manner. On contrary, Barthélemy et al. [10] utilize a local computational board of a camera to fulfill real-time requirements in a local context, but the applicability over a widely spread system (and other verticals) is still left as an open question.

Cicirelli et al. [11] present an agent-based, distributed platform for managing a network of computing nodes, spread within a city. Computation is conducted at the edge as well as the cloud, which handles computationally demanding tasks. Their proposed platform focuses on the dynamic deployment of new computing nodes and software agents for addressing geographical challenges, allowing a certain level of mobility for the agents (e.g., people with carry-on devices or vehicles). However, while their work focuses on the design and overall edge-cloud architecture, we aim in this paper to propose an edge-native architecture as well as a distributed method for efficiently tackling spatio-temporal challenges.

Taleb et al. [12] propose a Multi-Access Edge Computing (MEC) based architecture where services follow the users, with increased mobility causes service migration between the edge operators along the way. However, with regard to the geographical and spatio-temporal challenges, it is not clear that such a hop between operators can always take place in a resource-efficient manner. Buildings and geographical features of the terrain can either decrease the quality of the connection or block it entirely, and spatio-temporally rapid, fast-moving other devices in the same environment can cause unexpected load for service providers and edge services. Thus, we use as a de facto starting point for our work the situation where edge clients and nodes of different capabilities roam freely in a geographically diverse environment. This perspective is also considered by Giardano et al. [13]—but with limited evaluation of parameters affecting or affected by the movement between edge servers—and our previous work where we considered a real-life use case highlighting the phenomena [23]. In this work, we analyze such an environment through a simulation study.

City-scale sensing and data analytics. Smart cities rely on IoT, big data, cyber-physical systems, and edge-cloud computing continuum technologies [28,33] to provide data not only for novel applications but also other key functionalities of the enlarging urban spaces, such as increasing urban sustainability [2]. However, sensor data collection is rarely enough to provide timely feedback, decision support and situational awareness. Rather, multi-phased data processing is required. Pre-processing, cleaning, data fusion, and interpolation techniques can need to be widely considered before even the first steps of ML/AI learning phases can be run. Considering these pre-pocessing steps alone—add to the actual model building and evaluation—makes analyzing the huge amount of urban data both 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 still usually cloud-based (see e.g. [34]) or only partially supported by edge computing environment [32]. Some studies do concentrate on data analysis on the edge-based or edge-cloud continuum platforms [35], but give on only limited focus on the distribution of interpolation and analytics models crucial for the pre-processing steps and data distribution into the system.

Interpolation. There is a long history of interpolation based on Gaussian process (GP) regression [24]. A fundamental problem, however, is the method's computational complexity, relative to N^3 in processing time and N^2 in memory capacity, where N is the number of observations [24,36,37]. A few recent studies address this issue.

Some studies concentrate on the methodology of clustered Kriging. Park and Apley [38] present a method for patching together locally fitted spatial GP models by augmenting the data with pseudo-observations at the boundaries of the local models, such that the Kriging model remains formally a GP. Yasojima et al. [39] propose a heuristic approach using clustering, genetic algorithms and KNN for automatic estimation of variogram parameters in Kriging. van Stein et al. [36] propose a method for reducing the computational complexity of Kriging by partitioning the data set into smaller clusters with multiple Kriging models, and then applying approximative Kriging algorithms.

Further, Hernández-Peñaloza and Beferull-Lozano [40] present a distributed iterative Kriging algorithm for spatial interpolation in a wireless sensor network, where Kriging variance is reduced with iterative addition of new nodes to a cluster. The algorithm proposed by Chowdappa et al. [41] forms clusters of a wireless sensor network by minimizing Kriging variance and then estimates the semivariogram and interpolates locally in each cluster. Finally, Amato et al. [37] propose a spatiotemporal interpolation method, based on neural networks and centralized processing.

However, Yasojima et al. [39] do not aim to distribute the heavy computations related to GP regression, Hernández-Peñaloza and Beferull-Lozano [40] and Chowdappa et al. [41] only consider spatial interpolation, while Amato et al. [37] only consider fixed (i.e., non-mobile) sensors. Further, none of the above studies consider an edge-native architecture to mitigate the distribution of computations and reduce the burden on the core networks.

Edge-native distributed learning. A number of recent surveys (see e.g., [22,42–45]) review edge-native machine learning and EdgeAI approaches. While most approaches focus on distributed learning of neural networks (see e.g., federated distillation by Jeong et al. [46], a variant of Google's federated learning [47] approach), there are currently no neural network-based approaches for interpolation which can cope with mobile sensors [37]. Further, we have found no approaches which consider spatial covariance structures of the data in the distribution of learning.

3. EDISON

In this paper, we propose EDISON: a set of algorithms and an edge-native architecture for distributing spatio-temporal interpolation models, their computations, and the observed data vertically and horizontally between device, edge, and cloud layers. On the device layer, mobile and fixed sensors collect data, while IoT gateways provide connectivity and local data storage for the mobile sensors. The edge layer has edge servers (ES), placed at the fixed sensor locations, providing local computational capacity. Finally, the cloud provides centralized large-scale compute. An overview of the architecture is illustrated in Figure 1.

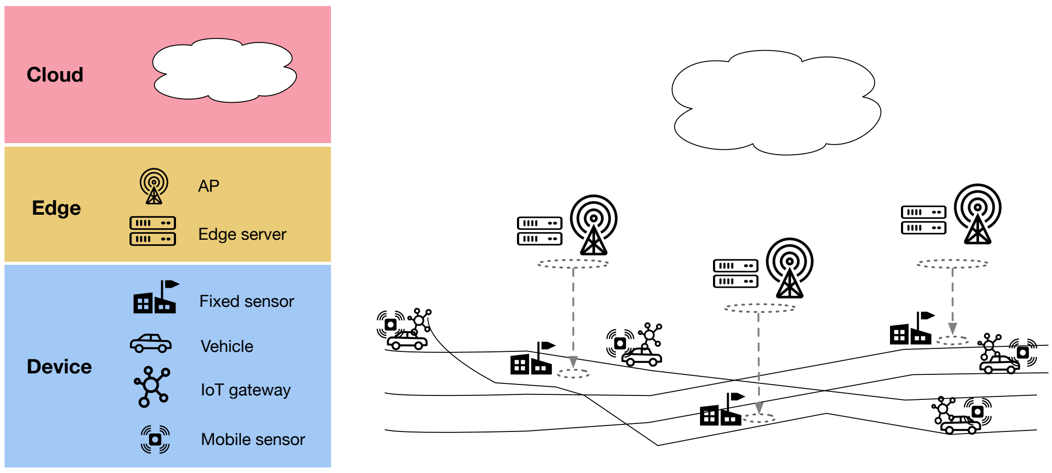

Figure 1. Overview of EDISON. The device layer comprises fixed sensors as well as mobile sensors mounted on vehicles. IoT gateways provide connectivity, store mobile sensor observations, and provide local computational capabilities. The edge layer enhances the fixed sensors with connectivity and further computational capacity. Cloud provides coordination and centralized processing.

EDISON assumes a small number of fixed sensors, e.g., radio weather sensors (RWS), and a massive fleet of mobile sensors mounted on vehicles. The mobile sensors use a short-distance wireless connection (e.g., Bluetooth Low Energy) to connect to an IoT gateway with a Wi-Fi uplink. Each fixed sensor is equipped with an edge server for local processing, as well as a Wi-Fi access point (AP), accepting connections from the IoT gateways on the vehicles, with sufficient range to cover vehicles passing by. The APs are assumed to be connected to a WAN with Internet connectivity.

The sensors are assumed to have, over a period of time, collected a spatio-temporal training set of the observed variables, and transmitted it to a centralized cloud server. The phenomenon predicted is assumed to be spatially distributed with relatively independent local data generating processes.

EDISON's operation comprises three distinct states, outlined below:

1. Calibration
   (a) CLOUD: Estimate calibration parameters for mobile sensors. Calibrate the collected sensor training set.
   (b) CLOUD: Transmit estimate calibration parameters to edge servers.
   (c) EDGE SERVERS: Transmit calibration parameters to IoT gateways passing by.
   (d) IOT GATEWAYS: Transmit calibration parameters to mobile sensors.
   (e) MOBILE SENSORS: Apply calibration.
2. Distributed learning
   (a) CLOUD: Partition the training set into subsets of observations around each edge server. Aim for subsets whose observations are maximally independent of the observations in other subsets.
   (b) CLOUD: Send the partitioned training set to all edge servers, rasterized to reduce transmission burden.
   (c) EDGE SERVERS: Fit a local, spatio-temporal interpolation model for the observations in the edge server's subset of the training set.
3. Distributed inference
   (a) MOBILE SENSORS: Send all observations immediately to the IoT gateway in the vehicle.
   (b) IOT GATEWAY: Store observations. Send stored observations to an edge server when passing by.
   (c) FIXED SENSORS: Send all observations immediately to edge servers.
   (d) EDGE SERVERS: Every time interval, find the right edge server (i.e., the right cluster) for each new mobile observation from IoT gateways that have passed by.
   (e) EDGE SERVERS: Send new mobile observations to selected edge servers.

(f) EDGE SERVERS: Every time interval, apply the local interpolation model with the data collected by the sensors.

The calibration and distributed learning states are employed only once, in the beginning of the operation. Having completed those states, the distributed inference state is the standard mode of operation (Figure 2). Calibrating the mobile sensors with a linear mixed-effects model and a rendezvous calibration model, based on the data provided by the fixed sensors, was proposed in our earlier study [23]. In this paper, we study the EDISON distributed learning and inference states. These are further detailed in the following subsections. The symbols used are listed in Table 1.

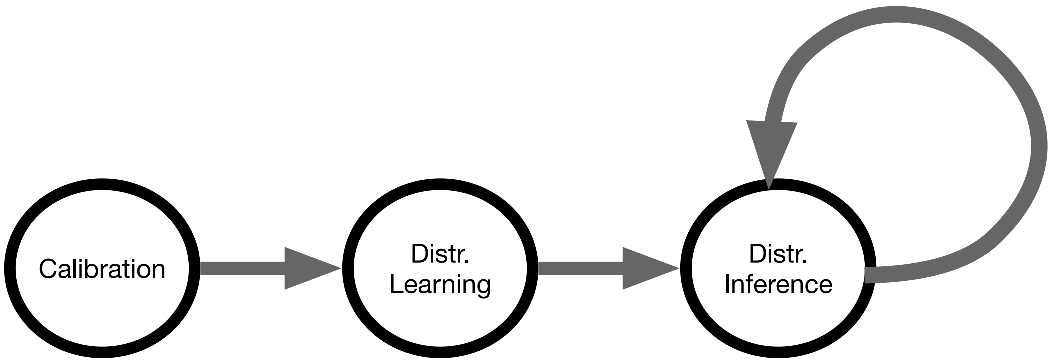

Figure 2. EDISON operational states. Calibration and distributed learning are employed once, in the beginning of operation, after which distributed inference is the standard operative state.

Table 1. Symbols in EDISON algorithms and equations.

Symbol	Range	Description
N	$\in \mathbb{N}$	the number of observations in the training set
L	$\in \mathbb{N}$	the number of observations for inference
M	$\in \mathbb{N}$	size of neighbourhood (i.e., n. of obs.) around each observation
K	$\in \mathbb{N}$	number of fixed sensors/clusters
$neigbourhood_i$		observations in the neighbourhood around observation i
O	$\in \mathbb{N}$	the number of raster cells on the map
$x_o,\ o \in [1,\ldots,O]$	$\in \mathbb{R}^2$	coordinates of the center of raster cell o
$f_l,\ l \in [1,\ldots,K]$	$\in \mathbb{R}^2$	location of fixed sensor l
Q	$\in \mathbb{N};$	the dimension of the interpolation model parameters
$\theta_i,\ i \in [1,\ldots,N]$	$\in \mathbb{R}^Q$	interpolation model parameters of the ngbh. around observation i
$\theta_o,\ o \in [1,\ldots,O]$	$\in \mathbb{R}^Q$	mean of the interpolation model parameters at raster cell o
$\theta_l,\ l \in [1,\ldots,K]$	$\in \mathbb{R}^Q$	mean of the interpolation model parameters at f_l
y_{ij}	$\in [0,1]$	membership of observation i to cluster j
λ	$\in [0,1]$	tradeoff between proximity and similarity in clustering
d	$\in \mathbb{N}$	size of neighbourhood for knn
z		the interpolation by the cluster model
$d(\cdot,\cdot)$	$\in [0,\infty]$	distance between two locations
$\{\cdot\}$		set

3.1. Distributed Learning

Distributed learning comprises two distinct phases, namely, (1) spatial partitioning of training set, taking place in the cloud; and (2) local interpolation model training, taking place on the edge servers. Figure 3 provides an overview of the process.

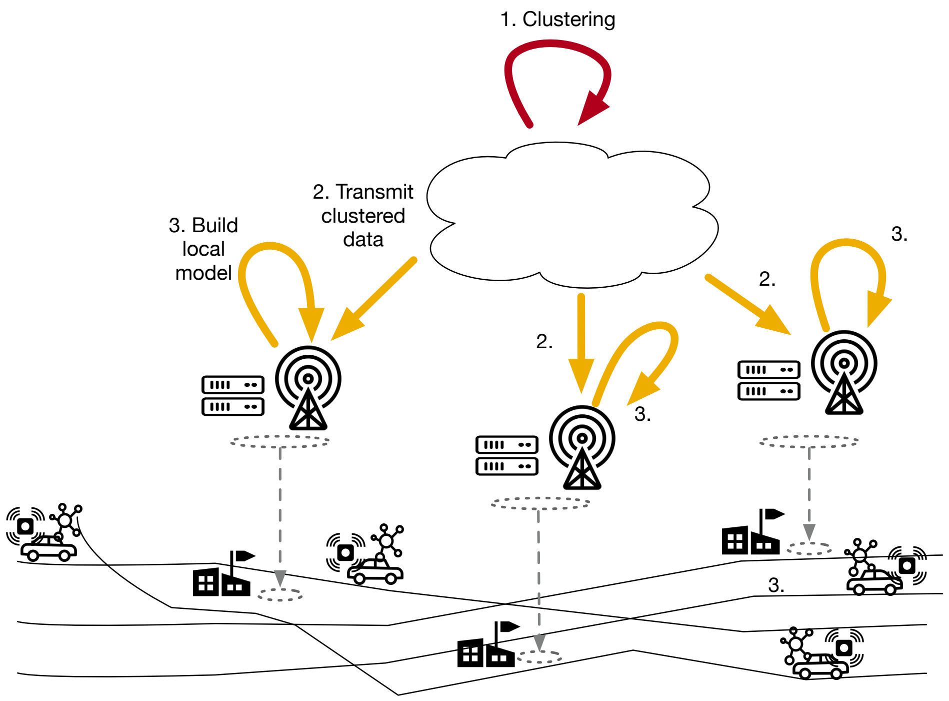

Figure 3. EDISON distributed learning. Cloud partitions the training set, the partitioned data is transmitted to the edge layer, and edge servers train local interpolation models.

Spatial partitioning of training set aims to divide the training set such that each edge server has an optimal subset of the observations, one subset for each edge server. Optimality here is based on the following intuitive and partially conflicting objectives:

- Independence: each subset should be as independent as possible from the others.
- Spatial connectedness: each resulting subset should be a spatially connected set of points.

Independence aims to maximize the overall quality of all the interpolation models, built by the edge server dedicated for each subset. Indeed, if the subsets of the training set are independent, the local models, trained with those subsets, can provide accurate interpolations with local data only. Independence thus aims for the quality of the local models individually, aiming for a partition that follows the spatial boundaries between the underlying data generating processes. Finally, spatial connectedness further drives the homogeneity of the subset while also making it easy to find clusters for new observations in the inference state.

EDISON aims for maximal independence and connectedness by emphasizing proximity and similarity. Proximity derives from Tobler's first law of geography, which states that everything is related to everything else, but near things are more related than distant things [48]. Similarity aims for the observations in a subset to vary in a similar pattern.

Algorithm 1 describes the distributed learning process formally. Similarity is based on the parameters of an interpolation model, fitted in the cloud for each observation with data in the spatial neighbourhood of that observation. Partitioning, conducted with a multidimensional clustering algorithm, aims to maximize the proximity and similarity of the training set subsets, based on the spatial coordinates of the observations and the parameters of the neighbourhood model for each observation.

The observations are rasterized on a grid of desired granularity before passing the raster for clustering. Setting the granularity, rasterization has two benefits. First, it reduces the computational complexity of the clustering, which is potentially computationally demanding with very large data sets, and second, it reduces the downstream data transmission burden when sending the clustered observations back to the edge servers.

Algorithm 1: Distributed learning

```
input  : observations $i \in [1, \dots, N], M, O, K, \{f_l\}_{l \in [1,K]}$, raster, $\{x_o\}_{o \in [1,\dots,O]}, d(\cdot,\cdot)$,
output: clustered observations
CLOUD:
    for each observation $i \in [1, \dots, N]$ do
        $neighbourhood_i \leftarrow M$ observations nearest to $i$;
        $\theta_i \leftarrow$ fit interpolation model for $neighbourhood_i$ ;
    end
    for each raster cell $o \in 1, \dots O$ do
        $\theta_o \leftarrow$ mean of $\theta_i$ in $o$;
    end
    for each fixed sensor $l \in 1, \dots, K$ do
        $\theta_l \leftarrow$ mean of $\theta_i$ at $f_l$;
    end
    clustered observations $\leftarrow$ find clusters$(\{f_l, \theta_l\}_{l \in [1,\dots,K]}, \{x_0, \theta_0\}_{o \in [1,\dots,O]})$;
    transmit clustered observations to edge servers ;
EDGE SERVER: on each edge server $l \in [1, \dots, K]$:
    $\theta_l \leftarrow$ fit interpolation model for observations in cluster $l$
```

3.2. Clustering

We propose a multidimensional spatial clustering method for EDISON. We base the method on our previous work on the PACK algorithm [49–51] modifying it for applicability in the EDISON environment. In more detail, PACK considers clustering an optimization problem, where the objective function for EDISON takes the following form:

$$\underset{y_{lo}}{\operatorname{argmin}} \sum_{l=1}^{K} \sum_{o=1}^{O} d(\{f_l, \theta_l\}, \{x_o, \theta_o\}) y_{lo}$$

with the following constraints:

1. $y_{lo} \in [0, 1] \quad \forall l, o$
2. $\sum_{l=1}^{K} y_{lo} = 1 \quad \forall i$

The constraints ensure a raster cell may belong to exactly one cluster (see [49,50]). The edge server locations f_l are constants, set to the locations of the fixed sensors.

We further apply a distance function

$$d(\{x_i, \theta_i\}, \{x_j, \theta_j\}) = \underbrace{\lambda \sum_{a=1}^{2} |x_{ia} - x_{ja}|^3}_{\text{proximity}} + \underbrace{(1-\lambda) \sum_{b=1}^{Q} (\theta_{ib} - \theta_{jb})^2}_{\text{similarity}},$$

where λ incorporates the trade-off between proximity and similarity in the clustering [50]. The proximity part is here cubed to ensure it dominates over long distances, keeping the clusters compact. $|\cdot|$ denotes the absolute value, required to keep the proximity part non-negative.

3.3. Distributed Inference

In the distributed inference state each edge server provides interpolations based on newly observed data and their locally trained models. Sensors transmit their observations to the edge server either directly, in case of fixed servers, or by way of the IoT gateway when passing by an access point, in case of the mobile ones. The edge servers, upon receiving new data from the mobile IoT gateways, partition the set of new observations according to the subsets of the training set found in the distributed learning state, and transmit those partitions to their designated edge servers (Figure 4).

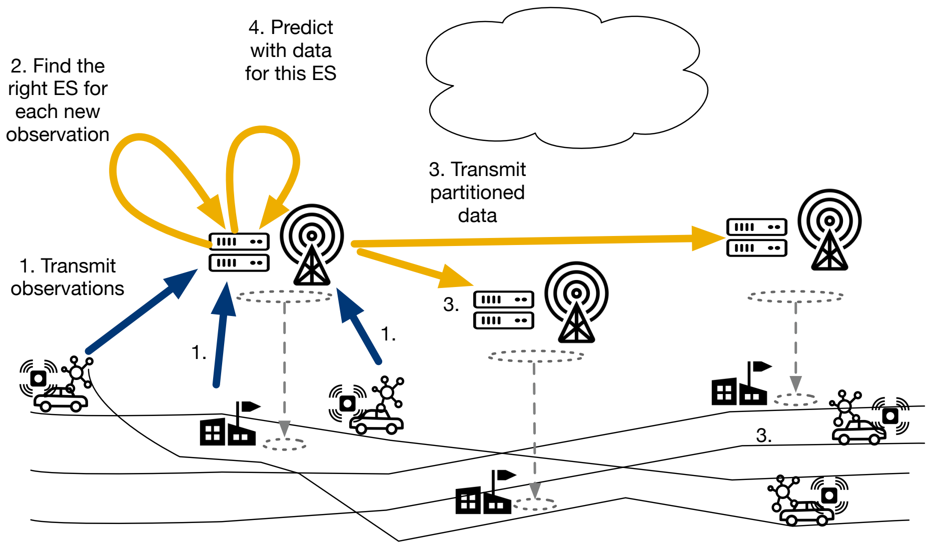
4. Predict with data for this ES
2. Find the right ES for each new observation
3. Transmit partitioned data
1. Transmit observations
1.
1.
3.
3.

Figure 4. EDISON distributed inference. Edge servers partition newly-observed data, transmit the partitions to their designated edge servers, and use the local new data for interpolation.

Algorithm 2 details the inference process, which comprises two distinct parts, processing in parallel. In the first part, the edge servers employ the k-nearest neighbours (knn) algorithm [52] to decide which edge server will be sent which new observation, and then transmit the observations over WAN to their designated edge servers. In the second part, the edge servers use the local models and the new local data, sent by the sensors and the other edge servers, to provide the interpolations.

Algorithm 2: Distributed inference

```
EDGE SERVER: on each edge server:
    input :new mobile observations $i^*$, clustered observations, distance function $d$
    clustered new observations ← knn($i^*$, clustered observations, $d$) ;
    transmit(new observations for cluster $l$ to edge server $l$)
EDGE SERVER: on each edge server:
    input :new observations $i^*$
    interpolate with new observations $i^*, \theta_p$ ;
```

4. Evaluation

We evaluate EDISON with simulated data to highlight the different complex spatio-temporal dependencies. The evaluation process involves the following steps:

1. Generate artificial ground truth data comprising complex spatio-temporal dependency structures.
2. Simulate sensor data.
    (a) Simulate static sensor locations.
    (b) Simulate mobile sensor trajectories.
    (c) Collect observations from the static sensor locations and along the mobile sensor trajectories.
3. Run EDISON.
    (a) Split the observations into training and test sets.
    (b) Conduct EDISON distributed learning on the training set.
    (c) Conduct EDISON distributed inference on the test set.
4. Calculate results.
    (a) Compare EDISON results to ground truth with RMSE.
    (b) Compare reference results to ground truth with RMSE.

Each step is described in detail in the subsections below.

4.1. Data Generation

We simulate a data-generating process, and a number of sensors observing it, on a rectangular 100×100 raster for 100 time-steps, comprising a total of 1 M data points. The data-generating process comprises four side-by-side spatio-temporal Gaussian point processes [24], independently affecting equal areas of a common map. Each process has a separable covariance structure $\Sigma = \Sigma_s \otimes \Sigma_t$, where $\otimes$ is the Kronecker product. The spatial component Σ_s is further set to follow the Matern covariance function, and the temporal component Σ_t the exponential covariance function. Finally, adding some Gaussian noise to the outcomes of the processes, we have

$$Y_{t,p} = a_p + \mathcal{X}_{t,p} + \epsilon, \quad \mathcal{X}_{t,p} \sim GP(0, \Sigma_p), \quad \epsilon_{p,t} \sim N(0, 0.5^2),$$

where $\mathcal{X}_{t,p}$ designates the spatial frames, one for each time step $t \in [1, T]$ and process $p \in \{1, 2, 3, 4\}$, a_p is a process specific intercept, and ϵ designates Gaussian noise with a standard deviation of 0.5. The spatial and temporal covariance matrices are generated with the R package `fields` [53].

Generating the data with a purpose-built R function, we further observe that for a linear combination $Z = LX$ of uncorrelated, normally distributed observations $X \in \mathbb{R}^N$, $X \sim \mathcal{N}(0, I_N)$, where L is a Cholesky factor of Σ (i.e., $\Sigma = LL^T$ is a Cholesky decomposition) and I_N an identity matrix of size N, we have for the covariance of Z

$$\mathrm{Cov}(Z) = \mathbb{E}(ZZ^T) = \mathbb{E}\Big((LX)(LX)^T\Big) = L\mathbb{E}(XX^T)L^T = LL^T = \Sigma,$$

where $\mathbb{E}$ is the expectation, and the penultimate equality derives from the non-correlation of the observations X. It thus suffices to generate uncorrelated, Gaussian observations X (with, e.g., the R `stat::rnorm` function), and multiply those with a Cholesky factor (with, e.g., the R `base::chol` function) of Σ.

The structure of side-by-side data generating processes reflects an urban environment where neighboring microclimates may vary considerably due to differences in, e.g., vegetation, heat sources, or construction materials and density [54]. We thus set a slightly different intercept a_p as well as covariance parameters for each of the four Gaussian processes (Table 2). The temporal covariance components all have range and phi parameters set as 1.0.

It could be argued that the ground truth, generated on a raster of 100×100 with 100 time frames, is too sparse to properly reflect a smart city environment. However, assuming a structure of (relatively) independent, side-by-side data generating processes, the 100×100 raster can be considered a randomly selected (and thus representative) region within the smart city, suitable for performance evaluation. Further, since we want to compare the results to a global interpolation, whose computational complexity is relative to N^3 [24], a raster of significantly finer granularity would be prohibitively heavy in computational burden. Furthermore, generating spatio-temporally correlated observations requires the Cholesky factorization of very large matrices (due to the Kronecker product), which is also not computationally feasible with finer granularity.

Table 2. Data generating process parameters for the spatial covariance components.

Region	a_p	Component	Cov. Funct.	Range	Smoothness	phi
1	12	Spatial	Matern	1	1.7	0.5
2	11	Spatial	Matern	9	0.7	2
3	15	Spatial	Matern	6	0.6	1.5
4	14	Spatial	Matern	0.5	1.7	0.1

4.2. Sensor Simulation

The simulated sensor network has 10 fixed sensors and a varying number of mobile sensors (Figure 5). The fixed sensors are located randomly across the area, while the mobile sensors start at random locations, at random timesteps, and follow a random walk trajectory with a step-lengt of 2 for 50 timesteps. Excess timesteps beyond the 100th are discarded. The fixed sensors provide observations every 2 timesteps, while the mobile sensors provide observations on every time step. Upon the termination of their trajectory, the mobile sensors are assumed to return to the nearest edge server to transmit their collected data. The mobile sensor trajectories are generated with the R package `trajectories` [55].

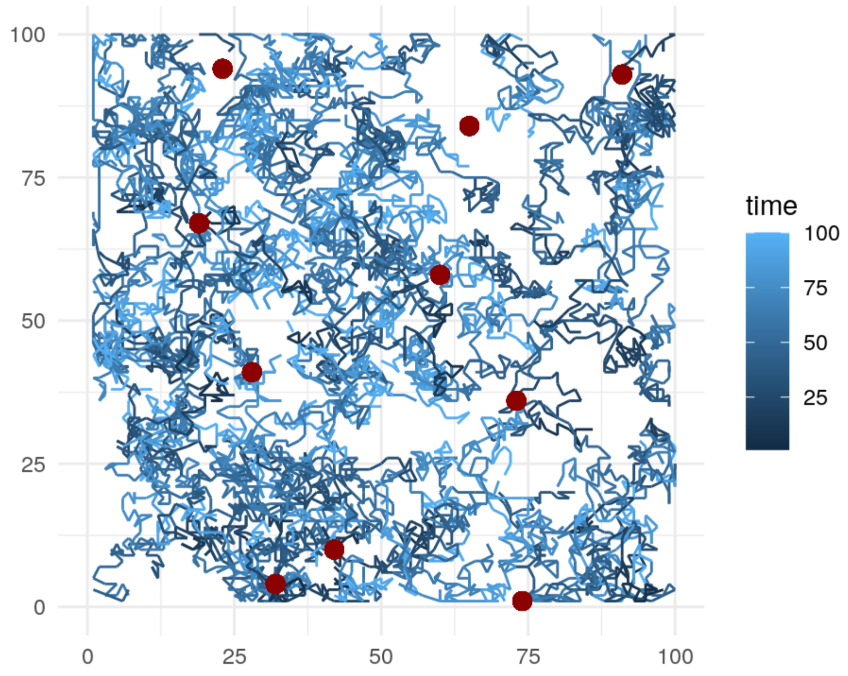

Figure 5. Simulated sensor trajectories. We marked 250 mobile sensor trajectories in blue, with the shade implying the time step. Fixed sensors marked in dark red.

4.3. EDISON

We split the data set 0.8:0.2 along the time axis, respectively, in a training set, used for EDISON distributed learning, and a test set, used for interpolation and comparison with the ground truth. We use Gaussian Process regression [24] for, respectively, fitting the pointwise spatiotemporal interpolation models and interpolating the new observations over the unobserved timeslots and locations. Fitting the Gaussian Process regression model assumes, here, only an intercept a in the deterministic component: $Y_t = a + \mathcal{X}_t + \epsilon$.

As the θ_i, that is, the similarity parameter for data point variation, we use the one-dimensional (i.e., $Q = 1$) variogram parameter sill (see e.g., [41]), estimated for each observation in a neighbourhood of $k = 80$ closest points. The pointwise variography and the subsequent clustering and partitioning are shown in Figure 6. The clustering parameter λ, capturing the tradeoff between the proximity and similarity of each point, is set to 0.001. Spatiotemporal variography and Kriging [24] employ the R `gstat` [56,57] and `spacetime` [58,59] packages.

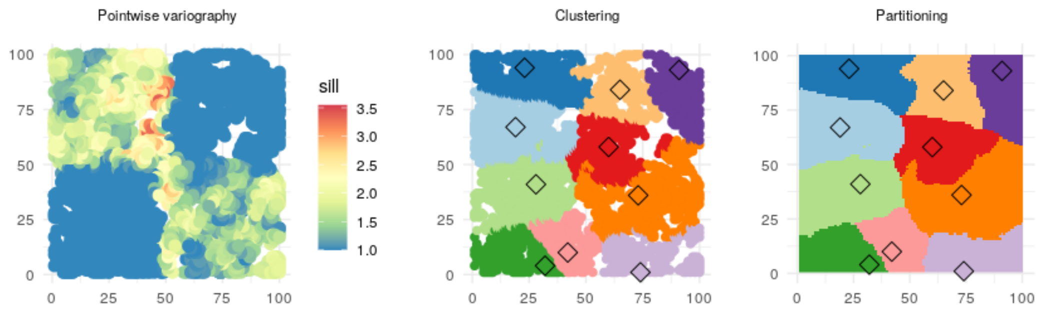

Figure 6. EDISON partitioning of data. The sill values of the pointwise variograms (**left panel**) clearly identify the boundaries between the four data-generating processes. Subsequent clustering (**middle panel**) finds those boundaries reasonably well. In the inference state, new observations can be partitioned among the clusters (**right panel**).

4.4. Results

We use the root mean square error (RMSE) to measure the quality of the interpolations in relation to the ground truth. We compare EDISON RMSE to that of some other possible approaches, listed below:

- global: unclustered interpolation over the whole map
- oracle: interpolation with pre-knowledge of the borders between the four data-generating processes
- baseline: each observation is assigned to the closest edge server
- E2: EDISON algorithm whose proximity part of the distance function (i.e., the spatial distance part) is squared, $d(\{x_i, \theta_i\}, \{x_j, \theta_j\}) = \lambda \sum_{a=1}^{2}(x_{ia} - x_{ja})^2 + (1-\lambda)\sum_{b=1}^{Q}(\theta_{ib} - \theta_{jb})^2$, instead of cubed (see Section 3.2)

The simulated ground truth, the observations, and the interpolations for oracle, baseline and EDISON can be seen in Figure 7, while a comparison of the RMSE values for EDISON, oracle, baseline and global for a varying number of mobile sensors are found in Table 3 and Figure 8. In short, EDISON outperforms the compared approaches. The results are further discussed in the following section.

Table 3. RMSEs of EDISON and the alternatives. Best results in each column highlighted in green.

Approach	**Mobile Sensors**		
	150	**250**	**300**
Global	1.30	1.19	1.14
Baseline	1.24	1.14	1.11
E2	1.25	1.15	1.14
EDISON (this study)	1.17	1.12	1.10
Improvement over global	10%	6%	4%
Improvement over baseline	6%	2%	1%
Improvement over E2	6%	3%	4%

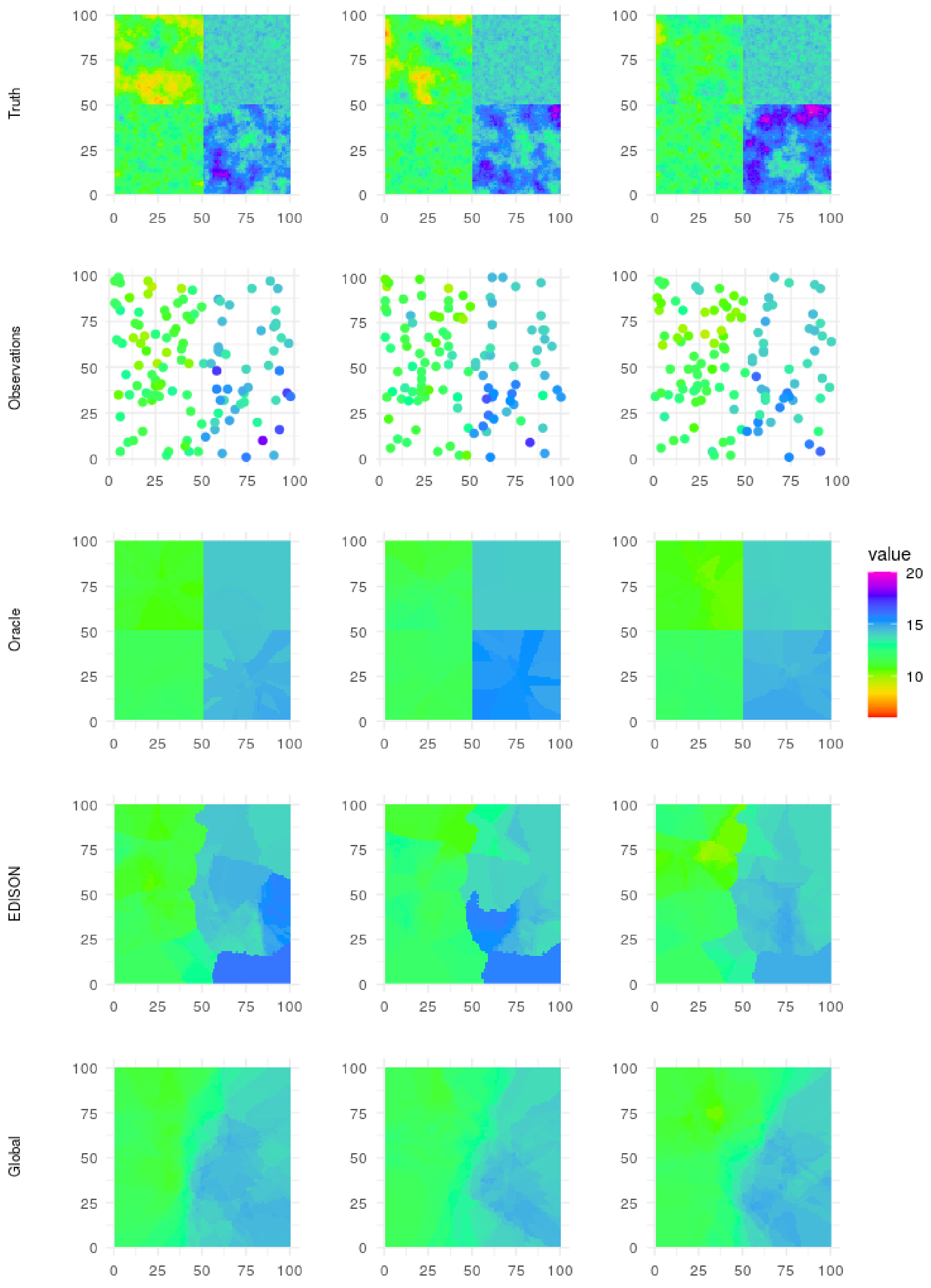

Figure 7. Ground truth, observations, and interpolations. The interpolations are conducted with oracle clustering, EDISON clustering, as well as a global variogram with no clustering. First three time frames of test data set are shown, from left to right.

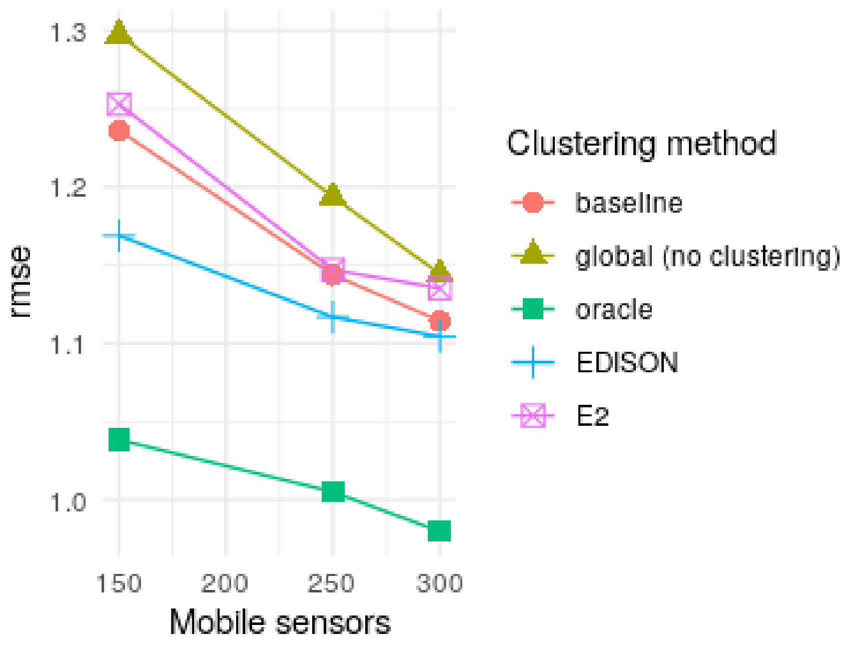

Figure 8. RMSE values. While interpolation with oracle clustering results in the lowest RMSE values, EDISON improves on both the baseline clustering as well as an unclustered, global interpolation.

5. Discussion

Results. The main motivation of EDISON is the distribution of the large-scale sensor data and the heavy computations related to spatio-temporal interpolation of the data. However, the RMSE results (Table 3 and Figure 8) show that in fact, even if global modelling were possible, EDISON would improve on the global interpolation by, at best, ca. 10%. Indeed, fitting a single variogram over the whole area and subsequently using that variogram for interpolation loses the detail of the local spatial processes and leads to worse overall performance.

Further, the RMSE results show that taking into account both proximity and similarity (see Section 3.1) further improves, by 6% at best, on the baseline k-median clustering algorithm, which accounts only for the proximity of the data. In the same vein, using a cubed distance instead of the squared one proposed by Ruha et al. [50] improves on the result with at most 6%. The cubed distance function emphasizes the spatial connectedness (see Section 3.1) of the clustering, favouring proximity over similarity when observations are spatially distant.

Limitations. As evidenced by the evaluation results (see Table 3 and Figure 8), EDISON shines when data is generated by a number of complex, relatively independent, spatially distributed processes. Such processes arguably include, for example, short-term surface temperatures in urban environments with a number of independent heat sources as well as varying surface materials and densities. However, as a result of the distributed nature of EDISON, the interpolated values often have sharp edges between the different clusters (see Figure 7, EDISON row). If the data-generating processes vary smoothly over long distances, such sharp edges may not be desirable.

Further, the current architecture has the mobile IoT gateways passing the observations to the edge servers over Wi-Fi upon rendezvous. While the setup is easy to deploy, it also introduces some limitations. For example, depending on the mobility patterns of the mobile sensors, the rendezvous events may be too rare to support timely interpolations. This is especially true for client applications requiring real-time or near-real time data.

Finally, if the mobility patterns of the mobile sensors have large spatial variance, that is, wide areas have few observations while others have many, the resulting cluster structure may not be optimal to provide high-quality interpolations, as some clusters may have too thin training data. The current architecture does not consider changes in the number or mobility patterns of mobile sensors. If, for example, the number of mobile sensors grows significantly, the edge servers may need to be augmented with further computational capacity. On the other hand, if the mobility patterns change, there may be a need for another round of clustering and local learning.

Future considerations. There are a number of possible avenues for mitigating the above limitations. The sharp edges between cluster interpolations, if undesired in the application, could be addressed by modifying the interpolation method. For example, the patchwork Kriging method by Park and Apley [38] could replace the ordinary Kriging approach used here. Patchwork Kriging generates pseudo-observations along the boundaries between neighbouring clusters to tie their results smoothly together. The resulting communication burden between the edge servers would, however, need to be closely considered for such a change.

Ruha et al. [50] considered also upper and lower capacity limits in the PACK clustering algorithm. EDISON could employ such limits for (1) to ensure each cluster has sufficient data for learning the local interpolation models (lower limit), and (2) assuming the mobile sensor trajectories maintain their spatial density, to ensure that the edge servers have enough computational capacity for the interpolation (upper limit). As such, the lower limit would likely improve the quality of the predictions in cases where the mobile sensor observations have large spatial variation in their density, whereas the upper limit, while ensuring computational capacity, would only reduce the quality of the predictions.

Further, for a more real-time operation, a mobile network (e.g., 5G or beyond [60]) could provide easy and fast connectivity with local MEC servers, capable of taking the role of the EDISON edge servers. Such a setup would, however, require a rethinking of the EDISON cluster architecture and the data flow in the distributed inference state due to the different placement of the MEC servers as well as the near-constant connectivity offered by 5G (see Figure 9).

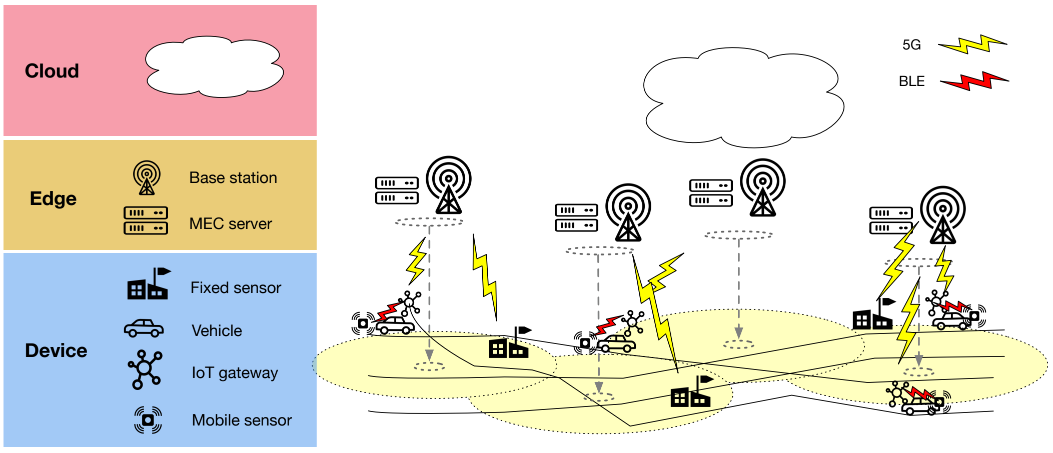

Figure 9. EDISON with Multi-Access Edge Computing (MEC). Coverage of each base station shown in yellow. Adapting EDISON for MEC requires a rethinking of the cluster architecture, based now around the BS locations. Further, due to the near-constant connectivity offered by 5G, data flow in the distributed inference state must be carefully reconsidered.

Finally, while the application here concentrates on interpolation, the same architecture could be used for predictive analytics in general. As future work, we plan to apply EDISON on various environmental sensor analytics topics such as local road surface temperature or friction prediction, extending our previous studies [61,62].

6. Conclusions

Smart cities aim, in part, to refine the observations provided by opportunistic, mobile sensor networks into fine-grained and reliable interpolations for city-scale contextual and situational awareness of the urban environment. However, the challenges of large-scale data, heterogeneous data providers, mobile and low-capability devices as an integral part of the system, and real-time requirements of many urban services, can make traditional cloud-based solutions infeasible. Indeed, few interpolation methods account for the computational complexity of current interpolation methods, and none consider also the communication burden caused by a massive sensor fleet uploading their observations over the wireless and fixed networks in the smart city.

This article proposed EDISON, an edge-native distributed AI architecture and a set of methods for interpolating the observations of a heterogeneous and sparse set of mobile and stationary sensors. EDISON addressed, in particular, smart city challenges related to large-scale data and mobile, low-capability devices. By partitioning the observations to clusters of manageable size, considering jointly the proximity and the similarity of the observations (see Section 3), EDISON trained local interpolation models over homogeneous data sets. The interpolation models, their computations, and the observed data were distributed vertically and horizontally between device, edge and cloud layers, minimizing both local computational burden as well as the communication load on the core network.

This study further included a controlled, simulated example with 1M observations of a spatio-temporal phenomenon. Compared to both a baseline solution considering only the proximity of the observations (“baseline”), a related solution considering both the proximity and the similarity of the results (“E2”), and a global, non-distributed solution, EDISON provided the lowest RMSE scores in all experiments.

Author Contributions: conceptualization, L.L., T.L. and L.R.; methodology, L.L., T.L. and L.R.; software, L.L. and T.L.; validation, L.L. and T.L.; formal analysis, L.L.; investigation, L.L.; resources, L.L., J.R.; data curation, L.L., I.L., L.R.; writing—original draft preparation, L.L., E.P., I.L.; writing—review and editing, L.L., E.P., I.L., L.R., M.J.S., S.P., J.R.; visualization, L.L.; supervision, S.P., J.R., M.J.S.; project administration, L.L.; funding acquisition, L.L., S.P., J.R., M.J.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by Academy of Finland 6Genesis Flagship (grant 318927), the Infotech Oulu research institute, the Future Makers program of the Jane and Aatos Erkko Foundation and the Technology Industries of Finland Centennial Foundation, and the personal grant for Lauri Lovén on Edge-native AI research by the Tauno Tönning foundation. Further, this research has received funding from the ECSEL Joint Undertaking (JU) project “FRACTAL: A Cognitive Fractal and Secure edge based on a unique Open-Safe-Reliable-Low Power Hardware Platform” under grant agreement No 877056. The JU receives support from the European Union’s Horizon 2020 research and innovation programme and Spain, Italy, Austria, Germany, France, Finland, Switzerland.

Data Availability Statement: All data used in evaluations were artificial. Instructions on data generation can be found in Section 4.1.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

AP	Access point
EDISON	Edge-native distributed interpolation
ES	Edge server
GP	Gaussian process
MDPI	Multidisciplinary Digital Publishing Institute

References

1. United Nations, Department of Economic and Social Affairs, Population Division. *World Urbanization Prospects: The 2018 Revision (ST/ESA/SER.A/420)*; United Nations: New York, NY, USA, 2019.
2. Meijer, A.; Bolívar, M.P.R. Governing the smart city: A review of the literature on smart urban governance. *Int. Rev. Adm. Sci.* **2016**, *82*, 392–408.

3. Gaur, A.; Scotney, B.; Parr, G.; McClean, S. Smart city architecture and its applications based on IoT. *Procedia Comput. Sci.* **2015**, *52*, 1089–1094. [CrossRef]
4. Strohbach, M.; Ziekow, H.; Gazis, V.; Akiva, N. Towards a big data analytics framework for IoT and smart city applications. In *Modeling and Processing for Next-Generation Big-Data Technologies*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 257–282.
5. Angelidou, M.; Psaltoglou, A.; Komninos, N.; Kakderi, C.; Tsarchopoulos, P.; Panori, A. Enhancing sustainable urban development through smart city applications. *J. Sci. Technol. Policy Manag.* **2018**, *9*. [CrossRef]
6. Naphade, M.; Banavar, G.; Harrison, C.; Paraszczak, J.; Morris, R. Smarter cities and their innovation challenges. *Computer* **2011**, *44*, 32–39. [CrossRef]
7. Lau, B.P.L.; Marakkalage, S.H.; Zhou, Y.; Hassan, N.U.; Yuen, C.; Zhang, M.; Tan, U.X. A survey of data fusion in smart city applications. *Inf. Fusion* **2019**, *52*, 357–374. [CrossRef]
8. Bokolo, A.J.; Majid, M.A.; Romli, A. A trivial approach for achieving Smart City: A way forward towards a sustainable society. In Proceedings of the 2018 21st Saudi Computer Society National Computer Conference (NCC), Riyadh, Saudi Arabia, 25–26 April 2018; pp. 1–6.
9. Jararweh, Y.; Otoum, S.; Al Ridhawi, I. Trustworthy and sustainable smart city services at the edge. *Sustain. Cities Soc.* **2020**, *62*, 102394. [CrossRef]
10. Barthélemy, J.; Verstaevel, N.; Forehead, H.; Perez, P. Edge-computing video analytics for real-time traffic monitoring in a smart city. *Sensors* **2019**, *19*, 2048. [CrossRef] [PubMed]
11. Cicirelli, F.; Guerrieri, A.; Spezzano, G.; Vinci, A. An edge-based platform for dynamic Smart City applications. *Future Gener. Comput. Syst.* **2017**, *76*, 106–118. [CrossRef]
12. Taleb, T.; Dutta, S.; Ksentini, A.; Iqbal, M.; Flinck, H. Mobile edge computing potential in making cities smarter. *IEEE Commun. Mag.* **2017**, *55*, 38–43. [CrossRef]
13. Giordano, A.; Spezzano, G.; Vinci, A. Smart agents and fog computing for smart city applications. In Proceedings of the International Conference on Smart Cities, Malaga, Spain, 15–17 June 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 137–146.
14. Deng, Y.; Chen, Z.; Yao, X.; Hassan, S.; Wu, J. Task scheduling for smart city applications based on multi-server mobile edge computing. *IEEE Access* **2019**, *7*, 14410–14421. [CrossRef]
15. Chiang, M.; Shi, W. *Grand Challenges in Edge Computing*; Technical Report; National Science Foundation: Washington, DC, USA, 2017.
16. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge Computing : Vision and Challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646. [CrossRef]
17. Kitchin, R. Making sense of smart cities: Addressing present shortcomings. *Camb. J. Reg. Econ. Soc.* **2015**, *8*, 131–136. [CrossRef]
18. He, Y.; Yu, F.R.; Zhao, N.; Leung, V.C.; Yin, H. Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach. *IEEE Commun. Mag.* **2017**, *55*, 31–37. [CrossRef]
19. Li, M.; Si, P.; Zhang, Y. Delay-tolerant data traffic to software-defined vehicular networks with mobile edge computing in smart city. *IEEE Trans. Veh. Technol.* **2018**, *67*, 9073–9086. [CrossRef]
20. Lovén, L.; Leppänen, T.; Peltonen, E.; Partala, J.; Harjula, E.; Porambage, P.; Ylianttila, M.; Riekki, J. EdgeAI: A vision for distributed, edge-native artificial intelligence in future 6G networks. In Proceedings of the 1st 6G Wireless Summit, Levi, Finland, 24–26 March 2019; pp. 1–2.
21. Partala, J.; Lovén, L.; Peltonen, E.; Porambage, P.; Ylianttila, M.; Seppänen, T EdgeAI: A vision for privacy-preserving machine learning on the edge. In Proceedings of the 10th Nordic Workshop on System and Network Optimization for Wireless (SNOW), Ruka, Finland, 1–4 April 2019.
22. Park, J.; Samarakoon, S.; Bennis, M.; Debbah, M.M. Wireless network intelligence at the edge. *Proc. IEEE* **2019**, *107*, 2204–2239. [CrossRef]
23. Lovén, L.; Karsisto, V.; Järvinen, H.; Sillanpää, M.J.; Leppänen, T.; Peltonen, E.; Pirttikangas, S.; Riekki, J. Mobile road weather sensor calibration by sensor fusion and linear mixed models. *PLoS ONE* **2019**, *14*, 1–17. [CrossRef]
24. Rasmussen, C.E.; Williams, C.K. *Gaussian Processes for Machine Learning*; The MIT Press: Cambridge, MA, USA, 2006. [CrossRef]
25. Lovén, L.; Peltonen, E.; Pandya, A.; Leppänen, T.; Gilman, E.; Pirttikangas, S.; Riekki, J. Towards EDISON: An edge-native approach to distributed interpolation of environmental data. In Proceedings of the 28th International Conference on Computer Communications and Networks (ICCCN2019), 1st Edge of Things Workshop 2019 (EoT2019), Valencia, Spain, 29 July–1 August 2019.
26. Iorga, M.; Feldman, L.; Barton, R.; Martin, M.J.; Goren, N.; Mahmoudi, C. *Fog Computing Conceptual Model*; Technical Report; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2018. [CrossRef]
27. Walravens, N. Mobile city applications for Brussels citizens: Smart City trends, challenges and a reality check. *Telemat. Inform.* **2015**, *32*, 282–299. [CrossRef]
28. Santana, E.F.Z.; Chaves, A.P.; Gerosa, M.A.; Kon, F.; Milojicic, D. Software platforms for smart cities: Concepts, requirements, challenges, and a unified reference architecture. *ACM Comput. Surv.* **2016**, *50*, 1–37. [CrossRef]
29. Mehmood, H.; Gilman, E.; Cortes, M. Implementing big data lake for heterogeneous data sources. In Proceedings of the 1st International Workshop on Data-Driven Smart Cities, in Conjunction with 35th IEEE International Conference on Data Engineering (ICDE 2019), Macao, China, 8–12 April 2019.

30. Raza, U.; Camerra, A.; Murphy, A.L.; Palpanas, T.; Picco, G.P. What does model-driven data acquisition really achieve in wireless sensor networks? In Proceedings of the 2012 IEEE International Conference on Pervasive Computing and Communications, PerCom 2012, Lugano, Switzerland, 19–23 March 2012; pp. 85–94. [CrossRef]
31. Peltonen, E.; Leppänen, T.; Lovén, L. EdgeAI: Edge-native distributed platform for artificial intelligence. In Proceedings of the 1st 6G Wireless Summit, Levi, Finland, 24–26 March 2019; pp. 1–2.
32. Hossain, S.K.A.; Rahman, M.A.; Hossain, M.A. Edge computing framework for enabling situation awareness in IoT based smart city. *J. Parallel Distrib. Comput.* **2018**, *122*, 226–237. [CrossRef]
33. Fortino, G.; Russo, W.; Savaglio, C.; Viroli, M.; Zhou, M. Modeling opportunistic IoT services in open IoT ecosystems. In Proceedings of the XVIII Workshop "From Objects to Agents", Scilla, Italy, 15–17 June 2017; pp. 90–95.
34. Baker, T.; Aldawsari, B.; Asim, M.; Tawfik, H.; Maamar, Z.; Buyya, R. Cloud-SEnergy: A bin-packing based multi-cloud service broker for energy efficient composition and execution of data-intensive applications. *Sustain. Comput. Inform. Syst.* **2018**, *19*, 242–252. [CrossRef]
35. Lagerspetz, E.; Varjonen, S.; Concas, F.; Mineraud, J.; Tarkoma, S. Demo: MegaSense: Megacity-scale accurate air quality sensing with the edge. In Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (MobiCom '18), New Delhi, India 29 October–2 November 2018; ACM: New York, NY, USA, 2018; pp. 843–845.
36. Van Stein, B.; Wang, H.; Kowalczyk, W.; Emmerich, M.; Bäck, T. Cluster-based kriging approximation algorithms for complexity reduction. *Appl. Intell.* **2020**, *50*, 778–791. [CrossRef]
37. Amato, F.; Guignard, F.; Robert, S.; Kanevski, M. A novel framework for spatio-temporal prediction of environmental data using deep learning. *Sci. Rep.* **2020**, *10*, 1–11. [CrossRef] [PubMed]
38. Park, C.; Apley, D. Patchwork kriging for large-scale Gaussian process regression. *J. Mach. Learn. Res.* **2018**, *19*, 1–43.
39. Yasojima, C.; Protázio, J.; Meiguins, B.; Neto, N.; Morais, J. A new methodology for automatic cluster-based kriging using K-nearest neighbor and genetic algorithms. *Information* **2019**, *10*, 357. [CrossRef]
40. Hernández-Peñaloza, G.; Beferull-Lozano, B. Field estimation in wireless sensor networks using distributed kriging. In Proceedings of the IEEE International Conference on Communications, Ottawa, ON, Canada, 10–15 June 2012; pp. 724–729. [CrossRef]
41. Chowdappa, V.P.; Botella, C.; Beferull-Lozano, B. Distributed clustering algorithm for spatial field reconstruction in wireless sensor networks. *IEEE Veh. Technol. Conf.* **2015**, *2015*. [CrossRef]
42. Park, J.; Wang, S.; Elgabli, A.; Oh, S.; Jeong, E.; Cha, H.; Kim, H.; Kim, S.L.; Bennis, M. Distilling on-device intelligence at the network edge. *arXiv* **2019**, arXiv:1908.05895v1.
43. Deng, S.; Zhao, H.; Fang, W.; Yin, J.; Dustdar, S.; Zomaya, A.Y. Edge intelligence: The confluence of edge computing and artificial intelligence. *IEEE Internet Things J.* **2020**, *7*, 7457–7469. [CrossRef]
44. Xu, D.; Li, T.; Li, Y.; Su, X.; Tarkoma, S.; Hui, P. A survey on edge intelligence. *arXiv* **2020**, arXiv:2003.12172.
45. Zhou, Z.; Chen, X.; Li, E.; Zeng, L.; Luo, K.; Zhang, J. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proc. IEEE* **2019**, *107*. [CrossRef]
46. Jeong, E.; Oh, S.; Kim, H.; Park, J.; Bennis, M.; Kim, S.L. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv* **2018**, arXiv:1811.11479.
47. Yang, Q.; Liu, Y.; Chen, T.; Tong, Y. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.* **2019**, *10*, 1–19. [CrossRef]
48. Tobler, W.R. A Computer Movie Simulating Urban Growth in the Detroit Region. *Econ. Geogr.* **1970**, *46*, 234–240. [CrossRef]
49. Lähderanta, T.; Leppänen, T.; Ruha, L.; Lovén, L.; Harjula, E.; Ylianttila, M.; Riekki, J.; Sillanpää, M.J. Edge computing server placement with capacitated location allocation. *J. Parallel Distrib. Comput.* **2021**, in press.
50. Ruha, L.; Lähderanta, T.; Lovén, L.; Kuismin, M.; Leppänen, T.; Riekki, J.; Sillanpää, M.J. Capacitated spatial clustering with multiple constraints and attributes. *arXiv* **2020**, arXiv:2010.06333.
51. Lovén, L.; Lähderanta, T.; Ruha, L.; Leppänen, T.; Peltonen, E.; Riekki, J.; Sillanpää, M.J. Scaling up an Edge Server Deployment. In Proceedings of the 2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), online, 23–27 March 2020; pp. 1–7.
52. Fix, E.; Hodges, J.L. *Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties*; Technical Report; USAF School of Aviation Medicine: Randolph Field, TX, USA, 1951.
53. Nychka, D.; Furrer, R.; Paige, J.; Sain, S. Fields: Tools for Spatial Data. R Package Version 11.6; CRAN. 2017. Available online: https://cran.r-project.org/web/packages/fields/index.html (accessed on 23 March 2021).
54. Dimoudi, A.; Kantzioura, A.; Zoras, S.; Pallas, C.; Kosmopoulos, P. Investigation of urban microclimate parameters in an urban center. *Energy Build.* **2013**, *64*, 1–9. [CrossRef]
55. McLean, D.J.; Volponi, M.A.S. trajr: An R package for characterisation of animal trajectories. *Ethology* **2018**, *124*. [CrossRef]
56. Pebesma, E.J. Multivariable geostatistics in S: The gstat package. *Comput. Geosci.* **2004**, *30*, 683–691. [CrossRef]
57. Gräler, B.; Pebesma, E.; Heuvelink, G. Spatio-Temporal Interpolation using gstat. *RFID J.* **2016**, *8*, 204–218. [CrossRef]
58. Pebesma, E. Spacetime: Spatio-Temporal Data in R. *J. Stat. Softw.* **2012**, *51*, 1–30. [CrossRef]
59. Bivand, R.S.; Pebesma, E.; Gomez-Rubio, V. *Applied Spatial Data Analysis with R*, 2nd ed.; Springer: New York, NY, USA, 2013.

60. Ahmad, I.; Shahabuddin, S.; Malik, H.; Harjula, E.; Leppanen, T.; Lovén, L.; Anttonen, A.; Sodhro, A.H.; Mahtab Alam, M.; Juntti, M.; et al. Machine Learning Meets Communication Networks: Current Trends and Future Challenges. *IEEE Access* **2020**, *8*, 223418–223460. [CrossRef]
61. Karsisto, V.; Lovén, L. Verification of road surface temperature forecasts assimilating data from mobile sensors. *Weather Forecast.* **2019**, *34*, 539–558. [CrossRef]
62. Lovén, L.; Gilman, E.; Riekki, J.; Läärä, E.; Sukuvaara, T.; Mäenpää, K.; Sillanpää, M.J.; Pirttikangas, S. Pilot study: Road–tyre friction prediction by statistical methods and data fusion. In Proceedings of the 2017 International Workshop on Smart Sensing System (IWSSS17), Oulu, Finland, 7–8 August 2017; University of Oulu: Oulu, Finland, 2017; pp. 1–2.