Pervasive and Mobile Computing 51 (2018) 27–42

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

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

Energy efficient distributed analytics at the edge of the network for IoT environments

Lorenzo Valerio [*], Marco Conti, Andrea Passarella

Institute of Informatics and Telematics, National Research Council, Via Moruzzi 1, Pisa, Italy

A R T I C L E   I N F O

Article history:
Available online 20 September 2018

Keywords:
Iot
Big data
Smart cities
Distributed learning
Communications efficiency

A B S T R A C T

Due to the pervasive diffusion of personal mobile and IoT devices, many "smart environments" (e.g., smart cities and smart factories) will be, generators of huge amounts of data. Currently, analysis of this data is typically achieved through centralised cloud-based services. However, according to many studies, this approach may present significant issues from the standpoint of data ownership, as well as wireless network capacity. In this paper, we exploit the fog computing paradigm to move computation close to where data is produced. We exploit a well-known distributed machine learning framework (Hypothesis Transfer Learning), and perform data analytics on mobile nodes passing by IoT devices, in addition to fog gateways at the edge of the network infrastructure. We analyse the performance of different configurations of the distributed learning framework, in terms of (i) accuracy obtained in the learning task and (ii) energy spent to send data between the involved nodes. Specifically, we consider reference wireless technologies for communication between the different types of nodes we consider, e.g. LTE, Nb-IoT, 802.15.4, 802.11, etc. Our results show that collecting data through the mobile nodes and executing the distributed analytics using short-range communication technologies, such as 802.15.4 and 802.11, allows to strongly reduce the energy consumption of the system up to 94% with a loss in accuracy w.r.t. a centralised cloud solution up to 2%.

1. Introduction

There is a unanimous consensus in the research and industry communities about the fact that IoT applications will account for a large share of the mobile data generated in the Internet [1]. Many reference market analyses (e.g. [2]), show that the adoption of IoT technologies will result in a huge economic impact in many sectors, well beyond ICT alone. As most of the value of IoT applications will come from the analysis of the data generated by IoT devices, the research area of IoT data analysis and management is a very exciting one.

The common trend in many current architectures [3] is to transfer IoT data from the physical locations where they are generated to some global cloud platform, where knowledge is extracted from raw data and used to support IoT applications. This is the case, among others, of the ETSI M2M architecture [4]. However, there are concerns whether this approach will be sustainable in the long run. The projections of the growth of the number of deployed IoT devices are exponential over the next years [2]. Mobile data traffic will grow at a compound annual growth rate (CAGR) of 47 percent from 2016 to 2021, reaching 49.0 Exabytes per month by 2021 [1]. Together with data generated by personal users' devices, this is likely to make the amount of data generated at the edge of the network huge, making it impractical or simply impossible to transfer them

[*] Corresponding author.
E-mail addresses: l.valerio@iit.cnr.it (L. Valerio), m.conti@iit.cnr.it (M. Conti), a.passarella@iit.cnr.it (A. Passarella).

https://doi.org/10.1016/j.pmcj.2018.09.004
1574-1192/© 2018 Elsevier B.V. All rights reserved.

to a remote cloud platform at reasonable costs. In addition, data might also have privacy and confidentiality constraints, which might make it impossible to transfer them to third parties such as global cloud platform operators. For example, in the manufacturing domain, data analytics is one of the cornerstones of the Industry 4.0 or Industrial Internet [5] concepts. However, most of the time industries will not be willing to move their data to some external cloud provider infrastructure, due to confidentiality reasons. On the other hand, they might not have the competencies and resources to build and manage a private cloud platform. Moreover, real-time delay constraints might require that data elaboration or storage is performed at the edge, i.e., close to where it is needed, rather than in remote data centres. Last but not least, as we show in this paper, sending data from IoT devices to remote data centres might result in very high energy consumption, even using wireless technologies designed for low power devices, such as NB-IoT. These trends and needs push towards a decentralisation of cloud platforms towards the edge of the network, where the paradigms of Edge computing, such as Fog [6], Mobile Edge Computing [7] and Cloudlets [8], can address these problems.

In this paper, we follow this approach and study the behaviour of a distributed learning solution based on Hypothesis Transfer Learning (HTL). In general, with HTL, instead of training a model on the whole training set in a centralised way, multiple parallel models are trained on disjoint subsets, and then the partial models are combined to obtain a single final model. We already have successfully applied HTL to the case of distributed learning in IoT environments in [9] and [10], where we have presented an activity recognition solution, where we have shown that such solution is able to drastically cut the network traffic required to perform the learning task, with an affordable reduction of accuracy with respect to a conventional solution where data is transferred on a global cloud platform.

Contrarily to [9,10], in this paper we consider that data, generated by IoT devices, need to be moved either to an edge gateway (and possibly to the cloud) or to a number of mobile nodes passing by the IoT devices. Both the gateway and the mobile nodes take the role of Data Collectors, and HTL is used to compute and exchange partial models between them. The main focus of this paper is on evaluating the interplay between the configuration of the distributed learning scheme and the involved network costs. This primarily depends on the assumptions about the spatio-temporal distribution of the nodes, and on the transmission costs for the various considered technologies. Specifically, we consider realistic wireless technologies for the communication between the nodes involved in the scenario, and we study the performance of different configurations of the HTL process in terms of the trade-off between the energy saving obtained by keeping data at the edge (and thus using short-range instead of long-range wireless technologies), and the loss of accuracy of the HTL scheme with respect to a conventional centralised learning scheme that can work on the entire set of data generated by all IoT devices. We compare different HTL configurations between them, and with a centralised solution where all data are sent to a remote cloud platform for processing. We compare different strategies of data collection, i.e. we go from the centralised solution in which data is collected by the edge server through cellular communication to using exclusively mobile nodes, in which devices collect all the data and perform the learning. We consider the most relevant wireless communication technologies, i.e., NB-IoT, LTE, 802.15.4 and 802.11. Precisely, we assume that IoT devices communicate with an edge gateway through energy saving cellular technologies (NB-IoT) and with mobile nodes using 802.15.4. Moreover, Mobile devices can communicate with each other through 802.11 and with the edge gateway using LTE.

This paper extends [11], as we take into account and evaluate in detail the impact of the specific wireless technologies used for communication, and we also consider the case where mobile nodes may lose data from IoT devices (while in [11] data collectors are only static nodes).

Changing the dataset would change the accuracy of our distributed learning scheme with respect to a centralised solution, which is something we have analysed extensively in [1–3] considering multiple datasets. However, for the specific focus of this paper, we think that focusing on one dataset already would be enough, as, thanks to our prior results, we can expect qualitatively similar results when the same analysis is performed on other datasets. We clarified this point in Section 1 of the paper.

To test the considered alternatives in a realistic setting, we selected a dataset related to an environmental monitoring application.[1] Precisely, the learning problem we consider is to learn a classifier able to predict the type of trees covering a delimited area of forest. We simulate an iterative data collection process interleaved by sessions of distributed learning on the newly collected data. However, for the specific focus of this paper, we think that focusing on one dataset already would be enough, as, thanks to our prior results, we can expect qualitatively similar results when the same analysis is performed on other datasets.

Our results show that it is possible to save up to 94% of energy in the system (mostly related to the data collection part) without relying on the edge servers with up to 2% of accuracy degradation with respect to a centralised cloud-based solution. Moreover, the computational burden of part of our solution can be significantly reduced with a negligible impact in terms of accuracy performance.

The rest of the paper is organised as follows. Section 2 presents related work. In Section 3 we introduce the problem and the reference scenario we consider in the paper. Section 4 describes the HTL-based distributed learning solutions used in our experiments. Section 5 describes the dataset and the performance metrics we used to perform the analysis presented in Section 6. In Section 7 we discuss the computational complexity of our solution. Finally, Section 8 concludes the paper.

[1] For the specific focus of this paper, we think that focusing only on one dataset would be enough, as, thanks to our prior results in [9–11], we can expect qualitatively similar results when the same analysis is performed on other datasets.

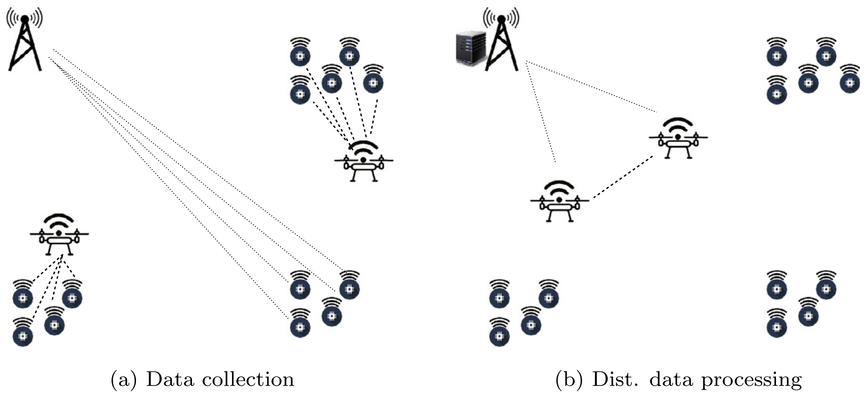


Fig. 1. Reference scenario.

2. Related work

The distributed learning problem can be approached in many different ways. One typical approach is represented by the so-called "ensemble methods" such as bagging, boosting and mixtures of experts [12–15]. Typically, these methods randomly split a dataset into smaller portions and allocate them to different classifiers that are independently trained. The individual classifiers are subsequently aggregated, sometimes after an additional training phase or with other techniques based on feedbacks coming from the training phase. Although these approaches are very powerful, they assume that the training set is available to a single coordinating processor.

Another very promising set of powerful solutions able to process huge amounts of data is represented by deep learning techniques [16–18]. These techniques are nowadays widely used to solve many complex tasks. Differently from our solution, these approaches do not target knowledge extraction where data have privacy constraints, or when network overhead should be minimised.

Other works propose fully distributed and decentralised learning algorithms. To the best of our knowledge, the more relevant with respect to our reference scenario are the following. In [19] authors present a distributed version of Support Vector Machine (SVM). Authors of [20] propose a distributed learning algorithm for arbitrarily connected machines. Another similar solution presented in [21] propose two distributed learning algorithms for random functional link networks. Both are iterative solutions that in order to converge to a model have to repeatedly exchange their partial models' parameters.

Other approaches that are closer, at least in terms of methodology to the one proposed in this paper include [22], where the authors propose a framework called Federated Learning whose purpose is to accomplish a distributed learning task on several mobile devices without exporting data from them. Briefly, in their approach, they train a neural network by executing the learning algorithm on each device and collecting in a centralised fashion the updates of the model. It proves to be very effective but, differently from our paper, they do not take into account aspect connected to the amount of traffic generated by the learning task.

Approaches based on distributed learning are emerging in the pervasive networking literature, e.g., to address the activity recognition problem. For example, in [23,24] several transfer learning-based approaches exploiting deep learning solutions are applied to the activity recognition problem. However, their main focus is on finding the most effective way to transfer layers of a neural network in order to exploit that knowledge to ease the training in other similar domains.

Finally, in [25] we have proposed an analytical solution that exploits a reference gradient-based learning algorithm (i.e. SVRG) to identify the optimal trade-off between accuracy and network overhead. Precisely in that we find the optimal operating point of data aggregation on the nodes that guarantee a certain target model accuracy. Differently, in this paper, we consider an approach based on the Hypothesis Transfer Learning framework in which entire models are moved between nodes instead of single updates of a single model, computed in separate locations.

The purpose of all these solutions is to learn a model whose accuracy performance is close, or at least comparable, with a centralised algorithm with access to the complete dataset. Although the accuracy is a crucial evaluation term for such kind of solutions, also the energy consumption triggered by those algorithms is fundamental, if we consider a network constrained scenarios like the IoT. In particular, in this paper, we are interested in evaluating the behaviour of the Hypothesis Transfer Learning framework in a context where realistic network related constraints are imposed.

3. Problem statement and system assumptions

In this section we describe the reference scenario we consider in this paper, as shown in Fig. 1. We assume a wide area that has been disseminated with several types of IoT devices, such as temperature sensors, humidity sensors, etc. These sensors are battery powered and they have limited computational and storage capacity. Since they are resource constrained, all the data they collect from the environment has to be transmitted to other more capable devices, in order to be processed. In this paper, we consider a slotted data collection process, that is, the sensors sense the environment and store the data for a fixed amount of time. At the end of a generic slot, they must transmit the data (within a certain deadline) in order to clean their memory and make room for the next session of data collection. In our scenario, the sensors have two ways for transmitting the collected data. In the first one, if a mobile device, that from now on we will call SmartMule (SM), is in the communication range of the sensors, it collects data from them through a short-range wireless technology. Note that a SmartMule can be any kind of mobile device capable to collect, process and analyse data, e.g. an SM can be a drone equipped with a chip-set appropriate for processing data, or a smart-phone, according to the specific sensing problem. The second one is used when no SM has come in the communication range of the device by the stated deadline. In that case a long range transmission occurs in order to send data to an edge server (ES) located at the edge of the network. For simplicity, we will term both SMs and ES as Data Collectors (DC). From a technological point of view, for long range transmissions, sensors can use one of the energy saving cellular technologies, e.g. NB-IoT [26], while for short/medium range transmissions they exploit a communication protocol based on the IEEE 802.15.4 standard, e.g. Zigbee [27]. Fig. 1(a) shows the data collection possibilities, through short range technologies (dashed lines) or long range ones (dotted lines).

At the end of each collection session, all DCs collaboratively learn a model (through HTL) based on the newly acquired data. This model is used to update the model elaborated until the previous time slot. Therefore, over time DCs elaborate a model, which is incrementally refined through the data collected after each collection slot.[2] It is worth noting that in this scenario data has not a locality constraint, i.e. once arrived on the first SM, it can be further aggregated on other SMs or even on the ES, according to the data management policy associated to the learning process. We assume that data can either be elaborated directly on the DC which collect them from the IoT devices, or can be exchanged and grouped at a smaller number of DCs, which are the sole responsible for computing the model. This allows us to have flexibility on the number of DCs, and on the locations where models are elaborated. We will study the impact of this configuration parameters in Section 6. In our scenario, we consider that devices can communicate with each others in two ways. One way is through cellular technology such as LTE or LTE/A. In this case we are assuming that SMs are far from each others (long range communication). Conversely if the devices are close enough, they can communicate through a short/medium range technology, such as WiFi or Bluetooth.[3]

In order to make our scenario more realistic we consider that the number of SMs roaming in the area for collecting data may vary from slot to slot. In addition, not only the number of SMs but also the amount of data they collect might be different from each other. To this end, the number of SMs present in the area during a single collection session is drawn from a Poisson distribution. In addition, we consider that SMs have a different mobility patterns in the sense that some SM may have the chance to visit a greater number of sensors than others, leading to an uneven amount of data collected on DCs. Specifically, we model the number of data collected by DCs during a session with a Zipf distribution. In order to allocate data to DCs we use the following procedure. To each DC in the area we assign an ID which also corresponds to a raking value from 1 to N, where 1 is the highest ranking and N is the lowest one (note that N is the total number of DCs in the area, drawn from the Poisson distribution). According to the Zipf's law each ranking value has a different probability of being drawn, therefore, in order to assign data to DCs, for each datum we randomly draw one of the N DCs' IDs. The result of such process is that the distribution of data on DCs is unbalanced, i.e. the number of data on each device and their internal distribution is not uniform. Moreover, we assume that the data collected by DCs are homogeneous, i.e., all the sensors positioned across several separate physical locations, collect data (possibly in separate time instants) belonging to the same generating phenomenon. Therefore, although mules in separate collection windows might have only a partial view of the all available data, in the long run they have access, on average, to all the available information, either in form of raw data or in form of aggregate partial models.

In this paper the learning task that all the DCs have to solve in a collaborative and distributed way is a classification problem. Specifically, we assume that each data partition located on a DC is a training set used to train a local classifier. The features of the local classifiers are exchanged with all the other DCs and used by each of the them to refine their own local classifier. These refined models are finally aggregated all together in order to combine their knowledge into a single "global" model, which is the one updated after each session. We point out that the model we learn from data is a linear classifier, that is the one of the simplest and the most lightweight models that can be learnt in terms of number of parameters. Nevertheless, its simplicity does not affect the generality of our analysis, because selecting a different model would affect only the quantity of data exchanged between DCs and not the ranking of the several configurations that we present in the rest of the paper.

Our purpose is to evaluate the performance of such system both in term of accuracy of the model learnt and the associated cost, defined as the energy spent for the wireless communications.[4] Precisely, our aim is to evaluate the impact that the considered communication technologies have on the cost of each part such system and to identify, depending on the specific operational conditions of the system, what are the most energy efficient configurations of the HTL algorithm in terms of where data has to be moved across DCs, for a certain target accuracy of the model.

[2] The main assumption here is that the time between consecutive batches should be significantly longer than the time required to train the model based on the data in each batch. If this is not the case, the entire approach considered in the paper would not be useful, as the model would be constantly "behind schedule" with respect to data generation rate. In such cases, either the application would downgrade its sampling rate, or mules should be improved in order to compute more quickly. However, this would be a dimensioning problem that is beyond the aim of this paper.

[3] The concept of "close enough" may vary depending on the specific technology considered.

[4] In this paper we do not include in our analysis the energy count used for the computation. We consider it as an orthogonal problem that deserves a dedicated study because it depends on many factors such as, among others, the specifics of the reference computational architecture.

4. Distributed learning through hypothesis transfer learning

In this section we briefly describe the distributed learning solution we use in this paper, based on the Hypothesis Learning Framework (HTL).

HTL methods are machine learning algorithms through which it is possible to exploit the knowledge acquired by a model m_1 trained on a certain set of data D_1 to ease the training phase of a model m_2, trained on a set of data D_2, different from D_1. The purpose is to improve the accuracy of m_2 using the knowledge contained in m_1.

This kind of solution is typically used in situations in which the size of D_2 is much smaller than the size of D_1. In fact in this situations, without a technique such as HTL, the accuracy of m_2 would be strongly affected by the small amount of data available to train the model.

In this paper, in order to perform the analysis described in Section 3, we exploit the distributed learning solution proposed in [9,10] whose core learning mechanism is based on GreedyTL, i.e. one of the reference solutions in the HTL literature [28]. Note that we do not provide all the mathematical details of GreedyTL. The interested reader should refer to the original paper [28]. Instead, we describe the key points of two distributed learning procedures. The first one is the distributed learning solution proposed in [9,10], that here we adapt to the scenario presented in Section 3. The second one is a modified version of the former, where the key difference is that the second scheme significantly reduces the amount of data exchanged between the DCs, as we explain in the following of the section.

From now on the first scheme is called *All-to-all HTL* (A2AHTL), while the second one *StarHTL* (SHTL). The basic assumption, that is valid for both A2AHTL and SHTL, is that each of the L DCs holds a local set of data D_i, with $i = 1, \dots, L$ and each D_i is different from the others, i.e. $D_i \cap D_j = \emptyset$, $\forall i, j = 1, \dots, L$ with $i \neq j$. Note that, the local datasets differ from each others not only for the content but also in terms of size, i.e. $|D_i| \neq |D_j|$, $\forall i, j = 1, \dots, N$ with $i \neq j$.

All-to-all HTL

We present now the A2AHTL scheme, fully described in [9,10], and summarised here for the reader's convenience. The A2AHTL solution is composed by 5 steps that are performed by each DC. We refer to Algorithm 1.

Step 0. (lines 1–5) In the first step each DC trains a classifier on its local data. At this step there is not a unique choice for the learning algorithm to be used, it mostly depends on the specifics of the problem at hand. In this paper we use a Support Vector Machine, one of the most well known machine learning approaches. We selected it because of its good performance and simplicity for the purpose of the learning task considered in this paper. Note that if the learning task changes, the only difference would be the amount of data exchanged between nodes, while all the rest of the algorithmic details will remain the same. After this first learning step, each DC l_i holds a model $m_{l_i}^{(0)}$.

Step 1. (lines 6–8) After Step 0, each DC l_i sends its model $m_{l_i}^{(0)}$ to all the other DCs. At the end of this phase each DC holds L SVM classifiers, each one trained on different data.

Step 2. (line 9) Each DC performs a second learning phase on the same local data, using the GreedyTL algorithm. In this step, the purpose of GreedyTL is to train a classifier that includes the knowledge contained in the models $m^{(0)}$ sent by all the other DCs. Precisely, GreedyTL solves an optimisation problem in order to find the linear combination of models $m^{(0)}$ which maximises that prediction accuracy with respect to the local dataset. At the end of this step each DC has a new classifier $m^{(1)}$ that is the output of the GreedyTL algorithm.

Step 3. (line 10) After the second learning phase, there is another synchronisation phase where all models are sent to a unique DC, which can be any of the involved DCs.

Step 4. (lines 11–14) The DC that receives all the models as per step 3 aggregates all of them into a single model $m^{(2)}$. In this paper we calculate the average model $m^{(2)} = \frac{1}{L}\sum_{l=1}^{L} m_l^{(1)}$. In this paper we can calculate the average of the models because assume that all the DCs learn the same type of model (i.e. in this case a linear classifier). Note that depending on the specific type of models selected for the learning process it might not be possible to average the models' parameters. Therefore, in such case a different aggregation strategy needs to be adopted. Moreover, once computed the final average model, all the partial models computed at step 1 and held by each DC are no longer useful and can be removed in order to save storage.

Star HTL

The StarHTL solution is a simple variation of the previous one. It is meant for those situations where the amount of data at each DC is strongly unbalanced. For example, a typical case is when the size of the local dataset at many DC is very small (e.g. even less than the size of the model to be learnt) and there is one DC that holds the majority of the data. In this case, we can save network resources by reducing the amount of models to send over the network as explained hereafter. For the sake of clarity we refer to Algorithm 2

Step 0. (lines 1–5) This step is unchanged w.r.t. A2AHTL. Each DC computes the model on its local data. After this first learning step, each DC l_i holds a model $m_{l_i}^{(0)}$.

Algorithm 1: All-to-all HTL

```
1: Let be $L$ the number of data locations
2: Let be $l_i$ the ID of the $i$th location and $l_c$ the ID of the current location $c$.
3: Let be $m^{(j)}$ the local model at step $j$
4: Let be $\mathbf{X_c}, \mathbf{y_c}$ the training pattern and training labels for location $l_c$, respectively
5: $m_{l_c}^{(0)} = TrainBaseLearner(\mathbf{X_c}, \mathbf{y_c})$
6: SendModelToAll($m_{l_c}^{(0)}$)
7: $M$= ReceiveBaseModels();
8: $M \leftarrow M \cup \{m_{l_c}^{(0)}\}$
9: $m_{l_c}^{(1)} = GreedyTL(\mathbf{X_c}, \mathbf{y_c}, M)$
10: SendModelToCenter($m_{l_c}^{(1)}$)
11: if IsCenter(c) then
12:     $H$ = ReceiveGTLModels()
13:     $H \leftarrow H^{gtl} \cup \{m_{l_c}^{(1)}\}$
14:     $m^{(2)}$ = CombineModels($H$)
15: end if
```

Step 1. (lines 7–9) Election of the "centralDC". During this step all the DCs exchange with the others a previously selected index that will be used to elect the central node. In this paper we used, as index, the standard Information Entropy, defined as follows:

$$H = -\sum_{\forall k \in K} p(k) log_{|K|}(p(k))$$

where K is the set of labels and $|K|$ is the total number of labels in the learning problem at hand. Precisely, each DC computes the entropy on its local dataset and the DC with maximum entropy is the one selected to be the central DC. In this way we are selecting as centre the one with more information in its local dataset, that is, the one that is likely to train a more accurate $m^{(1)}$ model.

Step2. (line 16) All the DCs ,apart from the one selected to be the centre, send their $m^{(0)}$ models to the centre DC.

Step 3. (lines 10–13) The centre DC is the only one that performs the second learning phase on its data using GreedyTL. At the end of this Step, the central DC has a new model $m^{(1)}$ built on its local data and aggregating the knowledge coming from the other models.

Algorithm 2: Star HTL

```
1: Let be $L$ the number of data locations
2: Let be $l_i$ the ID of the $i$th location and $l_c$ the ID of the current location $c$.
3: Let be $m^{(j)}$ the local model at step $j$
4: Let be $\mathbf{X_c}, \mathbf{y_c}$ the training pattern and training labels for location $l_c$, respectively
5: $m_{l_c}^{(0)} = TrainBaseLearner(\mathbf{X_c}, \mathbf{y_c})$
6: i =ComputeCenterIndex()
7: SendIndexToAll(i)
8: cId=ReceiveAndSelectCenter()
9: SendCenterIdToAll(cid)
10: if IsCenter(c) then
11:     $M$= ReceiveBaseModels()
12:     $M \leftarrow M \cup \{m_{l_c}^{(0)}\}$
13:     $m_{l_c}^{(1)} = GreedyTL(\mathbf{X_c}, \mathbf{y_c}, M)$
14: else
15:     SendModelToCenter($m_{l_c}^{(0)}$)
16: end if
```

It is clear that StartHTL generates less traffic than A2AHTL, as (i) models $m^{(0)}$ are not sent to all DCs, and (ii) there is no second synchronisation phase. Clearly, this might be paid in terms of lower accuracy.

5. Experimental setting

In this section we describe the dataset we used to perform our analysis and the indexes we used to measure the performance of the system in the considered scenarios.

Table 1
Nominal technical specifications of the considered wireless technologies.

Wireless Tech.	Tx P.(mW)	Uplink (Mbps)	Rx P.(mW)	Downlink (Mbps)
4G [29]	2100	75	2100	35
NB-IoT [30]	199	0.2	199,52	0.2
IEEE 802.15.4 [31]	3	0.12	3	0.12
IEEE 802.11g [32]	1080	48	740	48

5.1. Dataset description

The dataset we consider in this paper is connected to the forest cover type prediction problem (CovType).[5] It is a publicly available dataset that contains 581,012 observations. Each observation is a vector of 54 features containing both cartographic and soil information, corresponding to a 30×30 m area of forest. The learning task is to use this information to predict what is the main kind of tree covering the area. The dataset contains 7 classes. Since the number of observations for each class is not the same between classes, we performed a sub-sampling of the classes in order to remove such unbalancing. Thus, the final dataset contains 19,229 points ($\approx$ 2700 points per class). We used 80% of the dataset as training set and remaining 20% for test.

5.2. Performance metrics

In this paper we are interested in evaluating (i) the energy used to transmit data or models between IoT devices and DCs, and between DCs, in order to complete the learning task, and (ii) the accuracy of the model learnt through the distributed learning process. Note that we focus only on the energy used by devices for transmitting and receiving data or models using wireless technologies. We do not include in the energy count the energy spent by the edge server for transmitting and receiving data. Conversely we include the one used by devices for transmitting and receiving data to and from the edge server. The rational is that in this paper we are interested in analysing the impact of such system on battery powered devices.

In this paper, we use a simplified model in which, given a certain wireless technology, the specific power for transmitting and receiving data at a fixed data rate is constant for all the nodes in the network. We calculate the energy used for all the communications (both for transmitting and receiving data) as follows:

$$E = P * t \tag{1}$$

where P is the power, expressed in mW, for transmitting or receiving data and t is the duration, expressed in seconds, of the transmission. The duration of the transmission is calculated as follows

$$t = B/S$$

where B is the uplink (or downlink) data rate expressed in bit per second (bps) and S is the number of bits to be transmitted.

Note that P and B depend on the characteristics of the specific network technology. The ones considered in this papers are reported in Table 1. These are averages taken from reference papers found in the literature [29–32].

The overall energy used in the system for a session of data collection is the following:

$$E_S = E_C + E_L \tag{2}$$

where E_C is the sum of the energy used by the sensors to transmit data to the SmartMules or to the Edge server, while E_L is the energy used by the SmartMules during the distributed learning process.

In this paper, in order to evaluate the prediction performance of our system we use the well known F-measure [33]. The F-measure is a common aggregate index mostly used to evaluate the performance of a content retrieval system or a classifier. We use it because it provides a more precise description of the performance of our system with respect to the simple accuracy index defined as the number of correct predictions divided by the total number of predictions made. Precisely, through the F-measure we are able to summarise in one single index the performance expressed by the Precision and Recall indexes. The F-measure is defined as the harmonic mean between the precision index and the recall index, hereafter defined. Note that in this paper all the performance indexes are location-wise, i.e. they always refer to performance obtained by models trained in each separate location l.

The precision (or specificity) index is defined as the number of correct predictions divided by the total number of predictions made. More formally:

$$p_l = \frac{1}{n_l} \sum_{i=1}^{n_l} I(y_i, \hat{y}_i) \tag{3}$$

5 Dataset available at https://archive.ics.uci.edu/ml/datasets/Covertype

with

$$I(y_i, \hat{y}_i) = \begin{cases} 1 & \text{if } y_i = \hat{y}_i \\ 0 & \text{otherwise} \end{cases}$$

where $\hat{y}$ is the predicted class of the ith pattern x_i and y_i is its true class and n is the number of elements in the test set.

The recall (or sensitivity) index is defined as the number of correct predictions for a specific class divided by the number of the patterns that belong to that specific class, averaged over all classes. More formally:

$$r_l = \frac{1}{|C|} \sum_{\forall c \in C} \frac{1}{n_{l,c}} \sum_{i=1}^{n_{l,c}} I(y_{c,i}, \hat{y}_{c,i}) \tag{4}$$

Finally the F-measure is defined as follows:

$$F_l = 2 * \frac{p_l \cdot r_l}{p_l + r_l} \tag{5}$$

This measure takes values in the range [0, 1], where 0 stands for the worst prediction performance and 1 to the best one.

6. Results

We present now the results of our analysis. We recall that our interest is to identify through simulation how to configure the learning process such that, given a minimum target loss of accuracy (with respect to a centralised solution), the total amount of energy used to transfer data to the devices involved in the learning process is minimised. All the results we present hereafter are average values over 10 simulations. Confidence intervals, which we computed with a 95% confidence level, are very tight, therefore we do not show them, to improve the clarity of the plots.

Before presenting the results we provide the details that are shared by all the scenarios we analysed in this paper. As we explained in Section 3, we are considering a process of data collection through time. The process is divided in 100 time windows. In our simulations in each window are collected 100 observations, each one containing 54 features (the number of features is dataset dependent). At the end of each collection window, a learning phase takes place. The learning phase can be either distributed or centralised, depending on where the data has been sent. If all the data has been sent to the edge server, the learning phase will be centralised, while if the data is partially on the edge and partially (or totally) on the mules, the learning phase will be done using an HTL approach. Remember that regarding the last two cases, the learning algorithm is iterative and this is reasonable, as those nodes are sensing devices with limited storage, which typically have to flush their storage after a while. We remark that, although the specific dataset we use for evaluation the dynamic over time is slowly varying, we might expect that in cases where such a dynamic is quicker and data sensed in different time windows could not be that correlated, the learning performance would be different. However, we think that the results shown in the paper for the initial time slots provide indications of the performance obtained in these cases, where limited or no prior knowledge can be carried over between time slots. Remember that according to the scenario described in Section 3, in each collection interval, the number of mules that have collected data is a realisation of a random variable distributed according to a Poisson distribution of rate λ, while the number of data on each of such mules is distributed according to a Zipf law of parameter α. Specifically, we consider $\lambda = 7$ and $\alpha = 1.5$. We chose these parameters in order to simulate a scenario in which the data collection process is highly unbalanced, i.e. few SMs hold the vast majority of the data.

6.1. Benchmark scenario: all data on the edge server

Let us now present the analysis of the scenario that we use as benchmark during the rest of the paper. Precisely our benchmark is a scenario in which there are no mules in the area during the entire collection process. Therefore, for each collection session, all the data is sent to the edge server through NB-IoT, a cellular technology specifically designed for IoT devices. On the edge server, for each collection window the model is updated according to the new data. Results are shown in Fig. 2. In terms of accuracy the final F1-value that we obtain using a linear classification model is 0.63.[6] In absolute terms is not a particularly high value but is the best we can obtain from a linear model applied to this problem. However in this paper we are not interested in finding the best model that fits the data, but we are interested in analysing if through a distributed learning approach based on HTL it is possible to save resources while obtaining performance comparable to the one obtained executing the learning in a centralised fashion. Regarding the energy consumed for transmitting all the data using a NB-IoT we found a final value of 34,477 mJ that, for the rest of the paper we will use as benchmark in all the considered scenarios. Note that the values of accuracy and energy reported at the end of the collection process are equivalent to those obtained by the best performing centralised algorithm (in this case SVM) having access to the complete dataset.

[6] This final value corresponds to a centralised training session on the entire dataset.

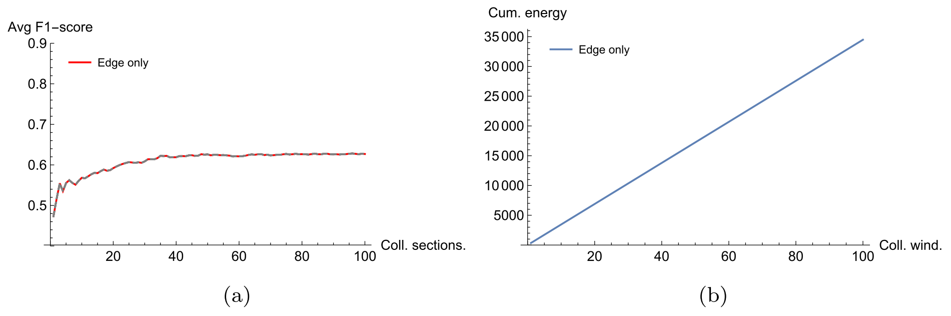


Fig. 2. Accuracy and cumulative energy cost for the edge only solution using NB-IoT.

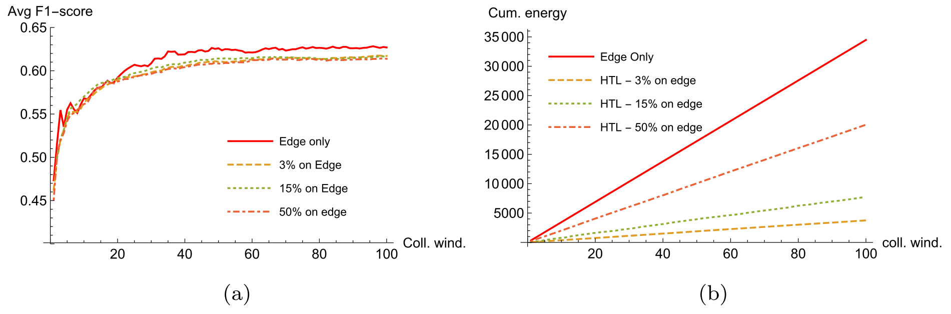


Fig. 3. Accuracy and cumulative energy cost for different percentages of data on edge.

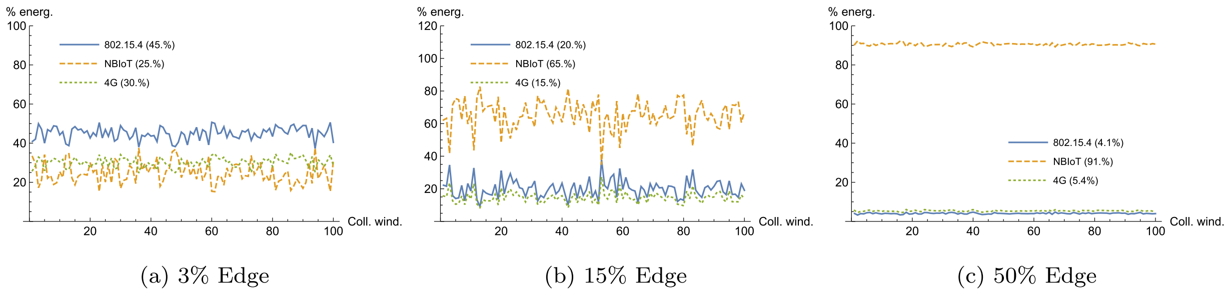


Fig. 4. Composition of the energy consumption for the tree scenarios.

6.2. Scenario 1: partial data on edge

In this scenario we want to evaluate if it is possible to save resources limiting the amount of data that is transmitted to the edge. To this end, we analysed tree separate cases in which we progressively decrease the amount of data sent to the edge server. Specifically we are simulating a scenario in which the SMs present in the area are able to collect only a portion of all the data produced by the sensors and the remaining one must be transmitted to edge in order to not be wasted. In this way, we are simulating a situation where no SMs are in proximity of some of the IoT nodes, thus the latter have to transmit their data to the Edge Server. In this scenario the IoT sensors use IEEE 802.15.4 for transmitting data to mules and if no mules are present, they use NB-IoT for sending data to the ES. Regarding the learning phase, in this scenario we present the performance accuracy obtained with StarHTL. We discuss only this method because in this specific scenario both of them obtain the same accuracy and StarHTL is lighter in terms of energy consumption. In this scenario, the SMs that are involved in the learning process communicate with each other and with the ES using 4G technology, i.e., we are allowing long range communications between the SMs.

As far as the energy consumption in concerned, we can see from Fig. 3 that collecting data through mules and sending the remaining data to the edge has the effect of saving energy resources. The reason of this is the fact that the transmission power of NB-IoT is greater than the one of 802.15.4, i.e. the former is 199 mW while the latter is 3mW. This makes NB-IoT more expensive than 802.15.4. It might look surprising that a technology like NB-IoT, which is designed to be energy efficient, turns out to be so energy hungry in our case. However, this is due to the fact that NB-IoT is designed having in mind short and infrequent transmissions. Our results show that it might not address IoT applications in general, as in cases

Table 2
Energy consumption and accuracy with different amount of data collected on the mules.

Scenario	Energy (mJ)	Gain (%)	F-Measure	Acc. Loss
Only Edge	34,477	0	0.63	0%
50% on Edge Server	20,047	42%	0.61	2%
15% on Edge Server	7736	77%	0.61	2%
3% on Edge Server	3749	89%	0.61	2%

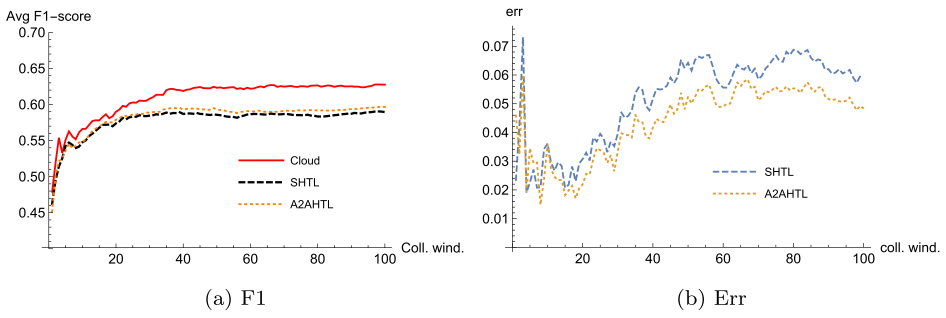


Fig. 5. Accuracy for A2AHTL and SHTL compared to the EdgeOnly solution, when no data is sent to the Edge. Initial data distribution is Zipf.

where significant amount of data have to be transferred from IoT devices, it becomes energy hungry (at least, with respect to short-range technologies). Therefore, introducing a short/medium range technology such as 802.15.4 for data collection is beneficial for the entire system. This is even more evident looking at Fig. 4b-c, which shows that, when the amount of data sent through NB-IoT becomes significant, the great part of the energy consumed for transmission is due to the data collection process. Conversely, the energy spent for the learning process and the short range data collection are comparable. However, looking at Fig. 4 we see that the data collection (which is done through IEEE 802.15.4 and NB-IoT) is more energy expensive than then the learning process (which is done through 4G), although, in absolute terms 4G is far more expensive that IEEE 802.15.4. The main reason for this behaviour is that StarHTL generates a very limited amount of traffic, proving it to be a very efficient and effective distributed learning solution. In fact in every scenario StarHTL is able to obtain an accuracy performance very close to the centralised one, i.e. the error with respect to the centralised solution is $\lesssim$ 2%. Note that this value is computed on the F1-score values obtained in the interval from the 50th to the 100th collection window, i.e. this interval is the one where the learning process converges.

In Table 2 we report the values of the energy consumed and the corresponding gain with respect to the Edge-only configuration, when a different amount of data is collected by mules instead of the edge server. Reducing the amount of data transmitted provides a saving of energy up to 89%.

6.3. Scenario 2: no data on edge server

From the previous results we found that not only it is more convenient from the energy consumption point of view limiting the amount of data to transmit on the edge server but it also allows to obtain learning performance comparable with a centralised solution. In this section we want to further stress this idea, completely removing the collection of data on the edge and assuming that all the data is collected by the SMs. The scenario at hand is the very same of the one considered before, i.e. the SMs receive through 802.15.4 all the data coming from the IoT sensors and after each collection window a learning session is performed. In this section, we compare the performance of the two learning mechanisms presented in Section 4, namely A2AHTL and StarHTL.

Looking at Fig. 5 we note that in this case, the loss of accuracy of both the HTL approaches passes from $\sim$ 2% of error to the $\sim$ 5% for A2AHTL and $\sim$ 6% for SHTL. This can be explained considering how data is distributed across the mules. We recall that according to the Zipf distribution, there is one mule with the vast majority of data (in this case one mule holds 55% of the data) while the others have the rest. Therefore in this scenario, although the number of mules in the area is on average 7, many of them hold just a very little portion of the existing data for each collection window. The immediate effect of this strong unbalancing is that the mules with few data observation will build inaccurate models and, since they are the majority, they do not positively contribute to the overall learning process. This is specially true for the SHTL approach, where only the mule designated as centre is the one that performs the retraining using the models from the other mules. In A2AHTL instead, the retraining is done by all the SMs, meaning that each of them uses the knowledge of the others to build the second model, and this has the effect of mitigating the problem arisen with SHTL.

Let us now look at Table 3 that contains the values of energy used by the collection and the learning processes. Regarding the latter, we compare two wireless technologies, i.e. 4G and IEEE 802.11g. Our aim is to investigate if the usage of short range wireless technology contributes to a further reduction of the energy consumption in comparison with the usage of

Table 3
Performance when all data is collected by the SMs.

Algo	Cum. En.(mJ)	Data Coll. (mJ)	Learning En. (mJ)	Gain (%)
A2AHTL - 4G	2789	1728	1061	91%
SHTL - 4G	**2513**	1728	785	**93%**
A2AHTL - wifi	5184	1728	3456	89%
SHTL - wifi	**2066**	1728	338	**94%**

Table 4
Performance when all data is collected by the SMs with data aggregation.

Algo	Cum. En.(mJ)	Coll. En. (mJ)	Learn. En. (mJ)	Gain (%)
A2AHTL - 4G	2957	1728	1229	91%
SHTL - 4G	2849	1728	1120	92%
A2AHTL - wifi	2700	1728	1028	92%
SHTL - wifi	2211	1728	483	94%

the 4G technology. We used IEEE 802.11g instead of the newer version of the protocol (i.e. IEEE 802.11n) because it is still the most diffused one. Before commenting the results it is important to specify that changing from 4G to Wifi, imposes us to modify the communication scheme between the SMs. In fact, we assume a star topology where one of the SMs acts as Access Point, initialising the wireless network to which all the other mules will connect and use to communicate. This is the configuration, for example, of WiFi Direct, which needs one device to act as Access Point for the others [34,35].

From the energy consumption point of view, looking at the first two rows (4G case) of Table 3 we see that using only the mules instead of using a combination of edge and mules for data collection and processing, further limits the costs. In fact, looking at the energy used by the two approaches we notice that SHTL is a more preferable choice the it is more energy efficient, i.e. we save 93% of energy w.r.t. the Edge Only solution, and the loss in accuracy with respect to A2AHTL is contained. In fact, compared to A2AHTL, SHTL is just 1% less accurate with respect to the centralised solution.

Considering the WiFi case, looking at the third and fourth row of Table 3 we notice the benefit of using IEEE 802.11g in combination with SHTL. In fact in the A2AHTL case, we notice that the energy spent for the learning process increases w.r.t. the one used by 4G. We justify this result by recalling that although IEEE 802.11g uses less transmission power, it has half of the downlink data rate of 4G. Moreover with the star topology managed directly by the mules, the number of transmissions made by battery powered devices increases. Conversely, for SHTL is even more convenient using IEEE 802.11g instead of 4G. The reason is that SHTL limits the number of models to be sent by the mules, i.e. we recall that in A2AHTL each model is sent to all the others while in SHTL only to the mule acting as centre. In light of this results we can state that in these settings if the mules have the possibility to communicate with each other using a short/medium range wireless technology such as IEEE 802.11g, it is possible to save up to 94% of energy with respect to the Edge Only solution, with a loss in accuracy in the range of 6%. Interestingly, the configuration of the HTL framework may play a significant role: if the loss of accuracy of SHTL with WiFi (the most energy efficient configuration) is not acceptable, then it is better to switch to 4G for communication between the mules, as A2AHTL with WiFi is less energy efficient than both HTL versions with 4G.

One aspect that affected the results obtained in the scenarios considered so far is that some of the SMs collected too few observations. Therefore, for these specific nodes, it would be more energy efficient to transmit data (which are few) instead of the local model. To cope with this problem, we adopt a very simple heuristic that is based on the idea that it is better to send models only when the amount of local data is above a given threshold. In our case the threshold is set to twice the size of the model. The mules that have an amount of local data below the threshold, aggregate their data on one of them until the condition is satisfied. Only the node that receives the data takes part to the learning process. In this specific case, after having applied this rule, the number of nodes involved in the distributed learning process passes from 7 to 3.

Our heuristic is very simple, but it proves to be effective in terms of accuracy. In fact, looking at Fig. 6 we see that the accuracy is improves with respect to the HTL configuration without aggregation. Specifically, SHTL is 3% less accurate than the centralised solution while A2AHTL is only 2% less accurate. It is worth noting that we obtained the very same loss of accuracy obtained in the *Scenario 1* without accumulating data on the Edge Server. Moreover, regarding the energy used to obtain such results we notice that, SHTL is always more energy efficient and also in this case, using WiFi instead of 4G proves to be beneficial, i.e. 94% of energy saved using WiFi and 93% using 4G. Also, note that in this case we avoid the increase of energy consumption of A2AHTL with WiFi, thus making using WiFi always more energy efficient than using 4G (see Table 4).

6.4. Scenario 3: no data on the edge server. uniform initial data distribution

So far we presented results considering that the distribution of data collected by SMs is strongly unbalanced. Now we investigate if the conclusions drawn for the Scenario 2 are still valid when we impose a different initial distribution of data between the mules. To this end, we consider a data collection scenario where each mule collects almost the same amount of data. Precisely, the amount of data that each mule collects, in this scenario is uniformly distributed, i.e. the amount of data per mule is in the range of 14%. Note that the main difference with the previous scenario is that in this case there is not a

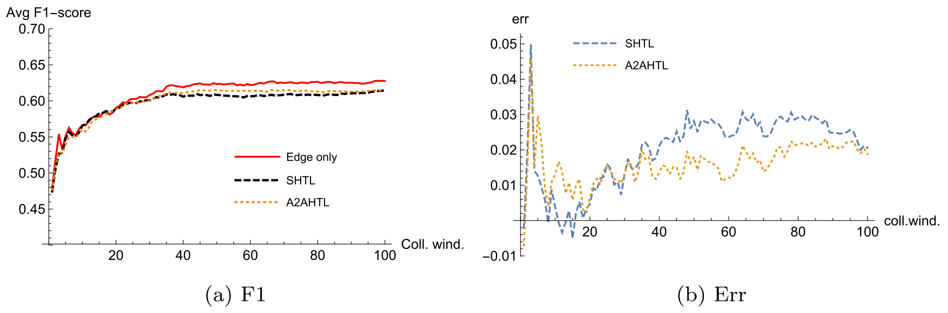


Fig. 6. Accuracy for A2AHTL and SHTL with "data aggregation heuristic" compared to EdgeOnly solution. Zipf data distribution.

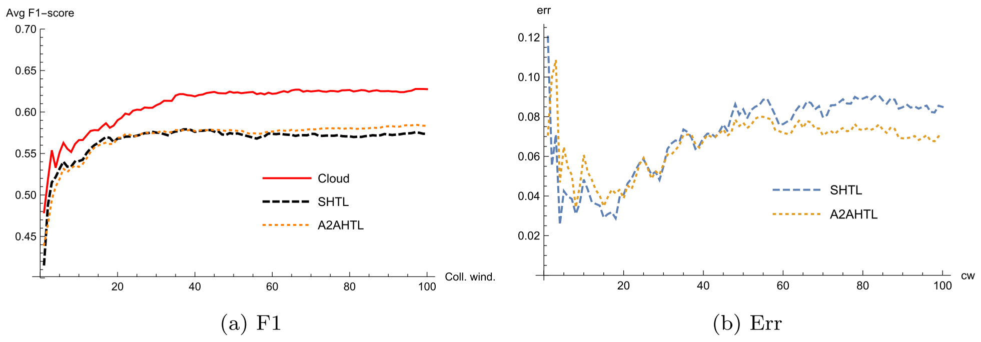


Fig. 7. Accuracy for A2AHTL and SHTL compared to the EdgeOnly solution. Uniform distribution.

Table 5
Performance when all data is collected by the SMs. Uniform distribution.

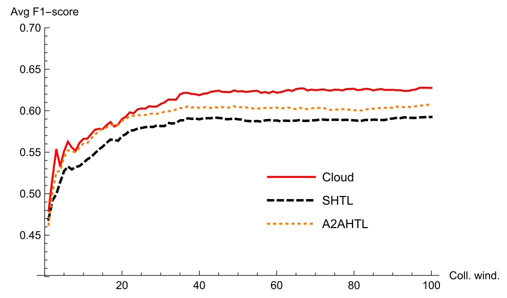
Algo	Cum. En.(mJ)	Data Coll. (mJ)	Learning En. (mJ)	Gain (%)
A2AHTL - 4G	3350	1728	1622	90%
SHTL - 4G	2753	1728	1025	**92%**
A2AHTL - wifi	6676	1728	4948	80%
SHTL - wifi	2169	1728	441	**94%**

single mule with more data than the others. Here, we performed the very same analysis done in Scenarios 1 and 2. In Fig. 7 the accuracy performance of both A2AHTL and SHTL, without data aggregation, are shown. Differently from the accuracy results presented in Scenario 2, we notice a degradation of performance, i.e. the loss in accuracy w.r.t. the centralised solution is in the range of 7% and 8% for A2AHTL and SHTL, respectively. This result is mainly due to the amount of data present at each SM. In fact since each of the SM has a very limited amount of local data (i.e. only 14 observations each) the initial models build at *Step0* of both A2AHTL and SHTL procedure under-fit the data, i.e. the models are not enough representative of the data on which they are built. Therefore the combination of these models results in a loosely accurate one. Instead, looking in Table 6 we notice the very same behaviour observed in Scenario 2, that is, SHTL is more energy efficient than A2AHTL and the wireless technology that allows to save more resources, between the ones considered in this paper, is IEEE 802.11g.

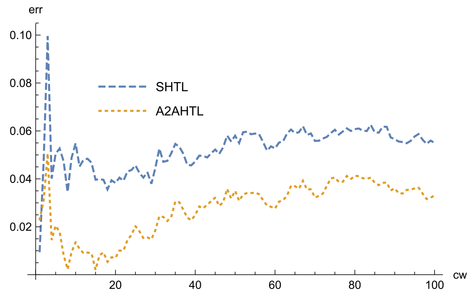
In order to evaluate if it is possible to reduce the loss of accuracy observed in the previous set of results, we apply also in this scenario the data aggregation heuristic, in order to increment the amount of data used by each mule to execute the distributed learning process. In this case, after the data aggregation process, the average number of nodes involved in the learning process passes from 7 to 3. Looking at Fig. 8, which shows the accuracy for both learning approaches, we see that SHTL obtains the best performance, reducing the loss of accuracy to 3% (see Table 5). Also in this final case, the data aggregation heuristic proves to be beneficial for improving the accuracy but, like in the previous case, at a marginal increment in terms of energy cost.

7. Computational complexity

Training a model could be computationally intensive and this might strongly affect the resource budget of battery powered devices. Specifically, the energy consumption related to computation depends primarily on the computational complexity of the part of the learning algorithm executed by mules. It would also depend on the specifics of the hardware and processing architecture of the mules, which would be dependent on the particular nodes used. Therefore, it is interesting



(a) F1



(b) Err

Fig. 8. Accuracy for A2AHTL and SHTL with "data aggregation heuristic" compared to EdgeOnly solution. Uniform initial data distribution.

Table 6
Performance when all data is collected by the SMs. Uniform distribution with data aggregation.

Algo	Cum. En.(mJ)	Data Coll. (mJ)	Learning En. (mJ)	Gain (%)
A2AHTL - 4G	3517	1728	1789	90%
SHTL - 4G	3235	1728	1507	**91%**
A2AHTL - wifi	3885	1728	2157	89%
SHTL - wifi	2377	1728	649	**93%**

to analyse the computational impact that our distributed learning procedure might have on such kind of devices. To make the analysis more general, we have (i) analysed the complexity of the algorithm implemented at the mules, and (ii) analysed the loss of accuracy when we reduce this complexity. Specifically, the complexity of the algorithm executed by mules, which is a function of the number of data points used for re-training the local models. We then analyse, via simulation, the impact on accuracy of reducing the number of data points. We show that the loss of accuracy is very little, even in case of a drastic reduction of the number of data points used. This shows that it is possible to significantly limit the energy spent in computation without losing significantly in terms of accuracy.

In our solution the two main computational demanding blocks are (i) the training of the base model (in our case a linear SVM) and (ii) the re-training using the GreedyTL algorithm. As shown in [36] the computational complexity to train a linear SVM is $O(N^2)$, where N is the size of the dataset. In our scenario we recall that the ith SM holds a portion of size n_i of the entire dataset, i.e., $N = \sum_{i=1}^{m} n_i$, where m is the total number of SMs, thus the computational effort, related to the SVM step, of the ith device is $O(n_i^2)$. This means that system-wise the computational complexity is $O(\sum_{i=1}^{m} n_i^2)$. This is by far less than $O(N^2)$, because it holds that: $\sum_{\forall i} n_i^2 < (\sum_{\forall i} n_i)^2$. Regarding GTL, for a single device, the complexity for training a model includes (i) the complexity for executing the $m-1$ source models, which in case of a linear model is $O(n)$ and (ii) the complexity for building the model which is, according to [37] $O(n^2)$. Therefore, the overall device-wise complexity is quadratic on the number of local data held by a device and, system-wise, it is $O(\sum_{i=1}^{m} n_i^2)$.

It is important to note that in our solution the SVM training can be considered as part of the problem definition, i.e., a design choice imposed to satisfy a constraint of the system. This means that, depending on the type of scenario, it could be replaced by another learning solution, with different computational requirements. Conversely, we analyse how the computational complexity of our solution changes when varying the configuration of GTL. Interestingly, as also shown in [28], one key feature of GreedyTL is that it proves to be very effective even when the amount of local data is small. This, can be exploited to significantly reduce the computational burden needed for training the model. For example, in [28,37] is shown that less than 10 data points per class are enough to train successfully a model. Therefore, as we show in the rest of this section, it is possible to control the impact of GreedyTL on the overall computational effort at a small cost in terms of accuracy.

To this end, we performed a set of simulations in which we incrementally decrease the number of data points per class provided to GTL. Precisely, we are interested in the relation between the number of data points used for the training (i.e., the operating computational complexity) and the accuracy obtained by the learning process.

In the following we present the results for all the cases considered in the paper, namely StarHTL and A2AHTL applied to Scenario 2 and 3. In all the considered scenarios, for each collection window there are 7 SMs (the same value used in the rest of the paper) and we evaluated both StarHTL and A2AHTL in which GreedyTL is trained using a random sample of size 2, 5, 10 data points, for each class. This quantities in the case of Scenario 3 (i.e., uniform data distribution) are equivalent to using, on average, the 15%, 40%, 75% of the data collected by each node, respectively. Conversely, since in Scenario 2 the data distribution is Zipf, such quantities are different from SM to SM. For the sake of clarity we reported in Table 7 the average percentages of local data used by each device during training. Note that SM1 is the device with more data and SM7 is the one with less data for each collection window.

In Figs. 9 and 10 we report the accuracy results obtained for the Scenarios 2 and 3. Each curve represents the accuracy of our approaches for different size of random samples used to train GreedyTL. As we can see, all the curves for both approaches

Table 7
Percentage of local data used by each device corresponding to the sub-sample sizes for Scenario 2.

Abs. Sample Size	SM1	SM2	SM3	SM4	SM5	SM6	SM7
2	4%	9%	18%	33%	40%	50%	100%
5	10%	22%	45%	83%	100%	100%	100%
10	19%	45%	90%	100%	100%	100%	100%

Table 8
Loss of accuracy compared to EdgeOnly for A2AHTL and StarHTL at different size of data used for retraining in Scenario 2 settings.

Algo.	$n = 2$	$n = 5$	$n = 10$	$n =$ All
A2AHTL	7%	6%	6%	5%
StarHTL	7%	7%	7%	6%

Table 9
Loss of accuracy compared to EdgeOnly for A2AHTL and StarHTL at different size of data used for GreedyTL training in Scenario 3 settings.

Algo.	$n = 2$	$n = 5$	$n = 10$	$n =$ All
A2AHTL	10%	8%	8%	7%
StarHTL	10%	8%	8%	8%

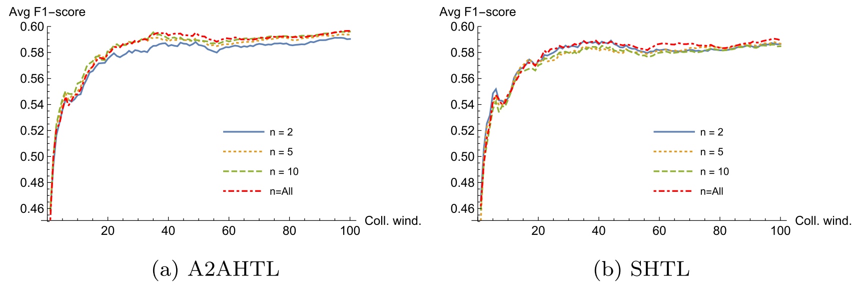


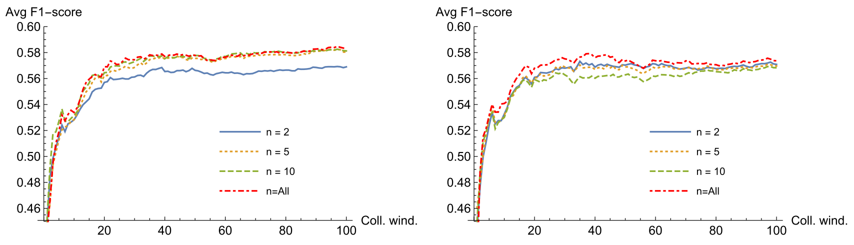
Fig. 9. Accuracy for different sizes of sample used by the re-training part. Zipf distribution of data on SMs.

are very close to each other. For the sake of comparison, we include in the plots also the accuracy curve obtained using all the local data to train GreedyTL (i.e., these curves are the same presented in the previous sections). Confidence intervals (95% level of confidence) have been computed but not shown they are too tight and would degrade the readability of the plots. In both scenarios we see that with SHTL the curves are very close to each other and most importantly, they are close to the one in which GreedyTL has been trained with all the local data. This means that we can obtain, as far as accuracy is concerned, the same results at a lower computational cost. Conversely, A2AHTL appears to be less robust than SHTL. In fact, with $n = 2$ in both scenarios the performance are inferior to the one corresponding to $n = 5$,10. The reason is that, for some SMs, 2 data points per class are not enough to identify which source models are really informative.

To have a quantitative estimation of the performance degradation induced by the amount of data used, in Tables 8 and 9 we report the mean accuracy values for all the approaches and sample sizes considered so far and we compare them with the values reported in Sections 6.3 and 6.4. In the settings of Scenario 2 (i.e., where the data collection process in unbalances and few mule hold the vast majority of data) GreedyTL allows us to limit the amount of data used in the learning process losing up to 2% of accuracy with respect to using all the data present in the local datasets. In Scenario 3, where data in uniformly distributed among mules, the loss of accuracy obtained by limiting the number of data observation used in the retraining phase is 3%. These results show empirically that, although at a relatively small cost, the proposed solutions can be, within certain limit, tuned to fit the computational requirements that might be imposed by a specific scenario.

8. Conclusions

In this paper we analysed the trade-off between accuracy and the energy consumed for collecting data from IoT sensors and performing a distributed learning task on such data. We identified through simulations how to configure the learning process in order to reach a target loss of accuracy with respect to a centralised solution, minimising the amount of energy used to transfer data to the nodes involved in the learning process. Our results show that, according to the network technologies we take into consideration it is possible to strongly limit the usage of cellular data transmission in favour of short-range wireless technologies. In fact, being able to manage the data collection (and aggregation) between mobile devices, in such a way that they can collect enough data, proved to be the solution that allows us to save up to 94% of energy



(a) A2AHTL (b) SHTL

Fig. 10. Accuracy for different sizes of sample used by the re-training part. Uniform distribution of data on SMs.

resources (compared to sending all the data using a low energy cellular technology such as NB-IoT) and to obtain a model with accuracy comparable with a centralised cloud based solution, i.e. with only up to 2% of loss of accuracy. Additionally, our approach is robust enough such that we can significantly control and reduce the computational impact of part our solution at a negligible cost in terms of accuracy (up to 3% with respect to the plain version).

Acknowledgements

This work is partly funded by the European Commission under the H2020 REPLICATE (691735), SoBigData (654024), and AUTOWARE (723909) projects.

References

[1] Cisco, Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015–2020, 2016, [online] http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html.

[2] J. Manyika, M. Chui, J. Bughin, R. Dobbs, P. Bisson, A. Marrs, Disruptive technologies: Advances that will transform life, business, and the global economy, McKinsey Global Institute, 2013, [online], http://tinyurl.com/zpkr7m6.

[3] E. Borgia, The Internet of Things vision: Key features, applications and open issues, Comput. Commun. 54 (2014) 1–31, http://dx.doi.org/10.1016/j.comcom.2014.09.008, URL http://www.sciencedirect.com/science/article/pii/S0140366414003168.

[4] ETSI TC M2M, ETSI TS 102 690 v2.1.1 (2013-10) – Machine-to-Machine Communications (M2M); Functional Architecture, 2013, http://tinyurl.com/hemc3po.

[5] H. Kagermann, W. Wahlster, J. Helbig, Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0, Tech. Rep., German National Academy of Science and Engigeering, 2013.

[6] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the Internet of Things, in: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, MCC '12, ACM, New York, NY, USA, 2012, pp. 13–16, http://dx.doi.org/10.1145/2342509.2342513.

[7] P.G. Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, E. Riviere, Edge-centric computing: Vision and challenges, ACM Sigcomm Comput. Commun. Rev. (2015).

[8] M. Satyanarayanan, P. Bahl, R. Caceres, N. Davies, The case for VM-based cloudlets in mobile computing, IEEE Pervasive Comput. 8 (4) (2009) 14–23, http://dx.doi.org/10.1109/MPRV.2009.82.

[9] L. Valerio, A. Passarella, M. Conti, Hypothesis transfer learning for efficient data computing in smart cities environments, in: International Conference on Smart computing (SMARTCOMP 2016), St. Louis, Missouri, 2016.

[10] L. Valerio, A. Passarella, M. Conti, A communication efficient distributed learning framework for smart environments, Pervasive Mob. Comput. 41 (2017) 46–68, http://dx.doi.org/10.1016/j.pmcj.2017.07.014, URL http://www.sciencedirect.com/science/article/pii/S1574119217303875.

[11] L. Valerio, A. Passarella, M. Conti, Accuracy vs. traffic trade-off of learning iot data patterns at the edge with hypothesis transfer learning, in: 2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), 2016, pp. 1–6, http://dx.doi.org/10.1109/RTSI.2016.7740634.

[12] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140, http://dx.doi.org/10.1007/BF00058655.

[13] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. System Sci. 55 (1) (1997) 119–139, http://dx.doi.org/10.1006/jcss.1997.1504, URL http://www.sciencedirect.com/science/article/pii/S002200009791504X.

[14] Y. Freund, R.E. Schapire, Y. Singer, M.K. Warmuth, Using and combining predictors that specialize, in: Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing, STOC '97, ACM, New York, NY, USA, 1997, pp. 334–343, http://dx.doi.org/10.1145/258533.258616.

[15] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, G.E. Hinton, Adaptive mixtures of local experts, Neural Comput. 3 (1) (1991) 79–87.

[16] A. Bordes, S. Ertekin, J. Weston, L. Bottou, Fast kernel classifiers with online and active learning, J. Mach. Learn. Res. 6 (2005) 1579–1619, URL http://dl.acm.org/citation.cfm?id=1046920.1194898.

[17] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q.V. Le, et al., Large scale distributed deep networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1223–1231.

[18] A. Coates, B. Huval, T. Wang, D. Wu, B. Catanzaro, A. Ng, Deep learning with COTS HPC systems, JLMR 28 (3) (2013) 1337–1345.

[19] A. Navia-Vázquez, E. Parrado-Hernandez, Distributed support vector machines, IEEE Trans. Neural Netw. 17 (4) (2006) 1091–1097.

[20] L. Georgopoulos, M. Hasler, Distributed machine learning in networks by consensus, Neurocomputing 124 (2014) 2–12, http://dx.doi.org/10.1016/j.neucom.2012.12.055, URL http://www.sciencedirect.com/science/article/pii/S0925231213003639.

[21] S. Scardapane, D. Wang, M. Panella, A. Uncini, Distributed learning for random vector functional-link networks, Inform. Sci. 301 (2015) 271–284, http://dx.doi.org/10.1016/j.ins.2015.01.007, URL http://www.sciencedirect.com/science/article/pii/S0020025515000298.

[22] J. Konečný, H.B. McMahan, F.X. Yu, P. Richtárik, A.T. Suresh, D. Bacon, Federated learning: strategies for improving communication efficiency, arXiv preprint arXiv:1610.05492, 2016.

[23] M. Oquab, L. Bottou, I. Laptev, J. Sivic, Learning and transferring mid-level image representations using convolutional neural networks, in: Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, IEEE, 2014, pp. 1717–1724.

[24] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks? in: Advances in Neural Information Processing Systems, 2014, pp. 3320–3328.
[25] L. Valerio, A. Passarella, M. Conti, Optimal trade-off between accuracy and network cost of distributed learning in Mobile Edge Computing: An analytical approach, in: 18th IEEE International Symposium on A World of Wireless, Mobile and Multimedia Networks, WoWMoM 2017, Macau, China, June 12–15, 2017, pp. 1–9, http://dx.doi.org/10.1109/WoWMoM.2017.7974310.
[26] NB-IoT, 2016, http://www.3gpp.org/news-events/3gpp-news/1785-nb_iot_complete.
[27] Zigbee, https://www.zigbee.org.
[28] I. Kuzborskij, F. Orabona, B. Caputo, Transfer learning through greedy subset selection, in: Proceedings of Intl. Conf. on Image Analysis and Processing, 2015.
[29] A.R. Jensen, M. Lauridsen, P. Mogensen, T.B. Sørensen, P. Jensen, LTE UE power consumption model: For system level energy and performance optimization, in: 2012 IEEE Vehicular Technology Conference (VTC Fall), 2012, pp. 1–5, http://dx.doi.org/10.1109/VTCFall.2012.6399281.
[30] R.S. Sinha, Y. Wei, S.-H. Hwang, A survey on LPWA technology: LoRa and NB-IoT, ICT Express 3 (1) (2017) 14–21.
[31] N. Ahmed, H. Rahman, M.I. Hussain, A comparison of 802.11ah and 802.15.4 for IoT, ICT Express 2 (3) (2016) 100–102, Special Issue on ICT Convergence in the Internet of Things (IoT), http://dx.doi.org/10.1016/j.icte.2016.07.003, URL http://www.sciencedirect.com/science/article/pii/S2405959516300650.
[32] K. Gomez, T. Rasheed, R. Riggio, D. Miorandi, C. Sengul, N. Bayer, Achilles and the tortoise: Power consumption in IEEE 802.11n and IEEE 802.11g networks, in: 2013 IEEE Online Conference on Green Communications (OnlineGreenComm), 2013, pp. 20–26, http://dx.doi.org/10.1109/OnlineGreenCom.2013.6731023.
[33] D.M.W. Powers, Evaluation: From precision, recall and f-measure to roc, informedness, markedness & correlation, J. Mach. Learn. Technol. 2 (1) (2011) 37–63.
[34] V. Arnaboldi, M.G.G. Campana, F. Delmastro, Context-aware configuration and management of WiFi direct groups for real opportunistic networks, in: 2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems, MASS, 2017, pp. 266–274, http://dx.doi.org/10.1109/MASS.2017.40.
[35] M. Conti, F. Delmastro, G. Minutiello, R. Paris, Experimenting opportunistic networks with WiFi Direct, in: 2013 IFIP Wireless Days (WD), 2013, pp. 1–6, http://dx.doiorg/.10.1109/WD.2013.6686501.
[36] S. Shalev-Shwartz, N. Srebro, SVM optimization: Inverse dependence on training set size, in: Proceedings of the 25th International Conference on Machine Learning, in: ICML '08, ACM, New York, NY, USA, 2008, pp. 928–935, http://dx.doi.org/10.1145/1390156.1390273.
[37] I. Kuzborskij, F. Orabona, B. Caputo, Scalable greedy algorithms for transfer learning, Comput. Vis. Image Underst. 156 (2017) 174–185, Image and Video Understanding in Big Data, http://dx.doi.org/10.1016/j.cviu.2016.09.003, URL http://www.sciencedirect.com/science/article/pii/S1077314216301370.