An Efficient Edge-Cloud Partitioning of Random Forests for Distributed Sensor Networks

Tianyi Shen, *Student Member, IEEE,* Cyan Subhra Mishra, *Student Member, IEEE,* Jack Sampson, *Member, IEEE*
Mahmut Taylan Kandemir, *Fellow, IEEE* and Vijaykrishnan Narayanan, *Fellow, IEEE*
The Pennsylvania State University, USA. (Email: {tqs5537, cyan, jms1257, mtk2, vxn9}@psu.edu)

***Abstract*— Intelligent edge sensors that augment legacy "un-intelligent" manufacturing systems provides cost-effective functional upgrades. However, the limited compute at these edge devices requires trade-offs in efficient edge-cloud partitioning and raises data privacy issues. This work explores policies for partitioning random forest approaches, which are widely used for inference tasks in smart manufacturing, among sets of devices with different resources and data visibility. We demonstrate, using both publicly available datasets and a real-world grinding machine deployment, that our privacy-preserving approach to partitioning and training offers superior latency-accuracy trade-offs to purely on-edge computation while still achieving much of the benefits from data-sharing cloud offload strategies.**

***Index Terms*—edge computing, random forest, edge-cloud partitioning, sensor network**

I. Introduction

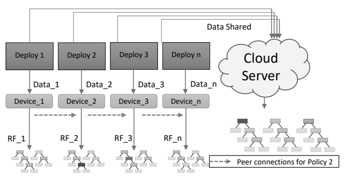
Retrofitting intelligent sensors nodes on legacy manufacturing systems provides cost-effective smart manufacturing upgrades. However, reliably meeting real-time analytics demands entirely within the limited compute and power budgets of these sensor nodes is challenging, especially for complex computational models such as DNNs. Therefore, simpler paradigms, like random forests, still remain popular for embedded sensors [7]. Additionally. techniques that balance communication and computation costs while partitioning the compute between the edge and a resource-rich server have been deployed. However, sharing data with as server makes data privacy a key constraint, especially when the server is an external service provider. Although model sharing, instead of data sharing, solves some of the challenges [5], such approaches are not trivial to deploy in classical learning paradigms, like random forests.

In smart manufacturing, multiple machines, even of the identical make and model, can generate different artifacts while encountering the same fault due to different physical interference such as resonant frequency and ambient temperature. The ability to capture diverse conditions from the different nodes, with or without sharing data, can lead to more robust models. We focus on extending random forests models that have been deployed in smart manufacturing [7] to explore edge-cloud partitioning strategies when multiple machines cooperate in contributing to better models. Constructing an accurate random forest model, while respecting data privacy of a distributed multi user sensor network is also challenging. Moreover, such a system demands accurately predicting the cases where the edge analytics were insufficient and the cloud must be employed for deeper analysis and accurate results.

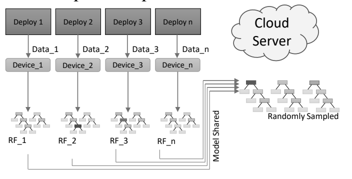
We propose a novel framework to perform intelligent edge-cloud partitioning for a distributed sensor network running random forest-based analytics. We propose novel inference strategies to maximize the number of predictions performed at the edge, while consulting the cloud only when the local results are not satisfactory. We also provide novel learning strategies, especially when the distributed sensors do not want to share the local data with the cloud, saving crucial communication latency and energy. Our contributions include: (1) Two different edge-cloud learning and inference policies, in a distributed sensor environment, to efficiently run random forest based data analytics. We explore the impact of privacy-preserving random forest training mechanisms to help protect sensitive data generated by the sensors. (2) Design of a threshold based edge-cloud partitioning policy which intelligently decides when to offload an inference to the cloud while maximizing the prediction accuracy and minimizing the communication overheads. (3) Evaluation of these policies on a publicly-available data set and also on data from real industrial grinding machines. We show that our privacy preserving partitioning approach outperforms edge-local prediction accuracy and achieves much of the accuracy in a data-sharing model. Finally, we provide a sensitivity analysis to understand the effect of different hyper-parameters on the accuracy and latency.

II. Edge-Cloud Partitioning Policies

In this section, we discuss the various policies to partition random forest compute efficiently between edge and cloud in a distributed sensor network. Each (edge) *device* (which is a part of the *Deployment* scenario like machine state monitoring) contains an embedded computer (e.g. raspberryPi) to sample, collect, and process the data and is also equipped with wireless communication to local/cloud server). The deployed edge devices perform the same analytics task (for example, monitoring the health of same type of machine at different sites) using a *random forest algorithm* (refer Algorithm-1). Since resource constraints can preempt complete execution at the edge, and sending data to the cloud is expensive due to communication energy and latency, it is essential to explore edge-cloud co-design, where we rely on the cloud if and only if it is necessary and



(a) Data Shared Policies. Red arrows indicates peer-to-peer connections



(b) Privacy Aware Policy: models are randomly sampled

Fig. 1: Edge-cloud partitioning policies.

beneficial to achieve a more accurate result. Furthermore, when the deployments are geo-distributed and operated by different owners, the privacy concerns on sharing the data with a third party cloud service provider becomes challenging in developing the analytics solutions.

A. Data-Shared Edge-Cloud Policy

First, we consider a distributed sensor model, where each device is equipped with a small compute capability to perform analytics using random forests, and each individual device is trained on its own data. Next, we consider a model where multiple devices from multiple deployments participate in cooperative data sharing. The cloud server, to accommodate data drift, periodically (but infrequently) accumulates all the data from different deployments to train a larger model which can generalize better than the local, edge specific, models. In both cases, due to the resource limitation of the edge devices, edge-specific model size may be reduced at the expense of accuracy.

Thresholding the Edge: The ideal case, for either of the aforementioned models, is when all the compute can be done, accurately, at the edge devices. However, to alleviate the shortcomings of less accurate predictions at the edge, some computations are routed to the cloud, expecting a better result at the expense of a higher latency. Therefore, it is essential to know if and when to route the compute to the cloud to balance accuracy and latency. We solve this issue by a adopting a thresholding mechanism based on model confidence, and accordingly decide to consult the cloud for better accuracy on less-confident local predictions. Deciding a proper threshold is application and quality of service dependent and remains a user tunable parameter. If the user task can tolerate lower accuracy prediction, or the user is more conservative about the cost associated with sending a request to the cloud, they could decide to change the threshold according to their requirements. Consequently, the programmable threshold makes the whole design more flexible: the collaboration ratio can be adjusted based on the needs of different types of machines, and more robust compared with the result dependent model simplification strategy which may perform differently on different datasets. Our experiments suggests variance to be the most suitable metric to decide the threshold.

Why Variance: Random-forest typically randomly samples the data and generates different estimators from the given data. The bootstrapping strategy makes each estimator learn different features of the data set and therefore increases the generality of the whole model. In this case, variance of the estimators becomes a good indicator of prediction confidence. A low output variance of the estimator means the predicted values are tightly concentrated, i.e., different estimators, even after learning different features, give similar answers. Similarly, a high variance indicates that the predicted values are discrete and there is no consensus. Relying on how different the answers from each estimators are, we can quantify the prediction quality of the random-forest.

Peer-Before-Server: Although sharing data with the cloud helps us build robust models, the communication latency is significantly higher than local computation, prompting several works [6] to push compute to the edge. For latency-sensitive applications, offloading to peers may be more viable than offloading to the cloud. That is, if the model at the edge node is unavailable (because of resource constraints) or has produced a low-confidence result, instead of directing the prediction query to the cloud, the edge could direct it to the peers (or other deployments) on the same local network. Since, all the deployments are working on the same task, and have trained on similar data, we will be able to perform the analytics task within a reasonable accuracy bound. However, if the confidence bound at the peer is not met, the cloud is contacted, at higher latency than having directly contacted the cloud to begin with. In our experiments, we observe that, in most cases, the accuracy of the peers are similar to that of the edge node. Therefore the case of consulting a peer only becomes beneficial when the communication latency and energy to the peer is much less than to the cloud.

B. The Case of Privacy Awareness

Having a centralized data repository of various systems improves model robustness. However, not all the deployments may want to participate in data sharing because of privacy reasons. Using the example of modern industrial machine state monitoring, the sensor attached to the machines for monitoring their health can also give away critical information like run time, operating conditions, materials etc., which might cause major privacy concerns. This problem has been addressed in federated learning by combining the models using weight averaging [2]. The algorithm makes sure that no data is shared with a centralized agent (like the cloud, which performs the training action), yet the learner is able to learn by combining multiple pre-learnt models. We extended this design idea by constructing a random forest model as a combination of multiple decision trees from different learners (different deployments). To preserve the data privacy of the participating deployments, the cloud learns by ensembling randomly sampled decision trees from each of the deployments, instead of learning a random forest from the data shared by each of them. Random sampling of the decision trees also augments the model by minimizing the data induced bias of each of the models. The random sampling method is data agnostic, and hence ensures minimal data induced bias. Random sampling of decision trees, albeit a naïve way, has been empirically shown to work well in preserving model characteristics and providing accurate predictions.

III. Experimental Evaluation and Results

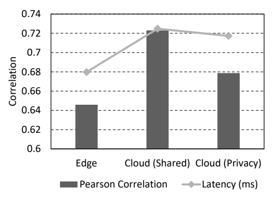
In this section, we describe our evaluation methodology and evaluate both privacy-preserving and data-sharing partitioning strategies compared with a traditional random-forest approach. We apply our techniques on Appliance energy prediction data [3], and also provide a case analysis of material surface roughness prediction using real world grinding-machine sensor data. We analyze the accuracy-latency trade offs of each strategy and show their benefits in different scenarios.

A. Appliance Energy Prediction

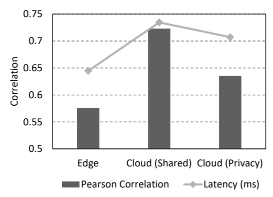
The appliance energy prediction data-set predicts the energy usage of home appliances, given the environmental parameters,

Algorithm 1: Training and Inference Pseudocode

```
function TRAIN(Sensor_Data, NodeID, CloudID)
    @Edge
    for each node do
        prep_data(data,node);          ▷ pre-process the data at edge
        if Privacy Aware then
            train_model()
                ▷ Locally train the model send_model(cloudID)
        else
            send_data(cloudID);        ▷ Send raw data to cloud
    @Cloud
    for each node do
        if Privacy Aware then
            sample_trees(nodeID);      ▷ Sample trees from each node
        else
            merge_data();              ▷ merge raw data from all nodes
        train();
end function
function INFERENCE(Data, NodeID)
    @Edge
for each node do
        Predict()
        if Accuracy ≤ Threshold then
            Send_data(CloudID);        ▷ Send data to cloud for
        accuracy
        else
            Send_results(CloudID);     ▷ Send the inference result
        Predict@Cloud                  ▷ Run Prediction at Cloud
end function
```



(a) *Data from 2 homes*



(b) *Data from 4 homes*

Fig. 2: Accuracy comparison of different policies on a simulated distributed setup. The data set is divided into 2 chunks creating a two home set up and the similar is done for a 4 home setup.

such as temperature and humidity of different regions of the home as well as the locality (from weather station data [3] with 14803 training samples and 4932 testing samples). This data set directly fits our use case for two reasons - 1. In the real world, these sensors would be distributed in different homes, and each home will have its own idiosyncrasies. 2. The home owners may or may not be willing to share the sensor measurements of their home for privacy reasons. We divide the data into training and testing sets, and to simulate a distributed environment, the training data is further divided into multiple different chunks (starting from 2 to 4, each part representing a household in the same neighbourhood). We ensure data diversity between different partitions to ensure similarities with the real-world. Further, we apply our policies to the distributed data and train random forest models (both for the edge and the cloud). Figure 2 shows the accuracy of various policies on 2-home and 4-home setups.

Following are the key observations from our experiments.

1) The data sharing with the cloud increases the accuracy of the power prediction significantly (11.94% increase in correlation for a 2-home setup and about 25.62% increase in

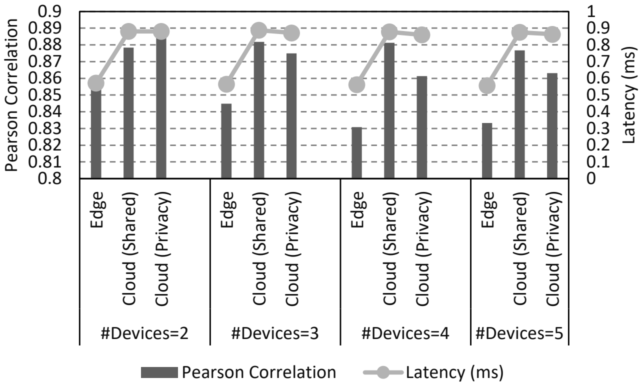


Fig. 3: Accuracy-latency comparison of different policies with different distributed setup. The edge execution is done on a raspberry Pi, and the cloud is mimicked by a desktop class machine. Considering the scale of the problem, the execution time on a desktop machine is almost same as a larger cluster that we typically find in a cloud.

correlation for 4-home setup). The accuracy improvement in the 4-home setup is significant and the reason for the lower edge accuracy of the 4-home setup is the limitation in number of training samples (2 home setup has twice the amount of data than the 4 home setup to train with).

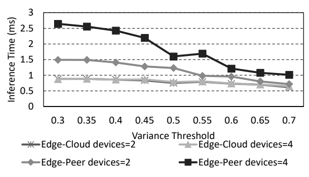
2) However, the cloud execution latency also increases for the 2-home setup due to the model complexity. The 2-home setup, albeit more accurate, has a more complex model (#splitting points 10501) at the cloud thanks to the large volume of training samples, leading to more execution time compared to the relatively simpler model for the 4-home setup. The cloud model is trained on a larger data set and has more parameters than the edge models and hence is more accurate.

3) The privacy preserving cloud model, although less accurate than the data shared cloud model, performs significantly better than the edge models, with 5.1% more correlation in case of the 2-home setup and 10.37% more correlation for the 4-home setup. Thanks to a simpler model (which is a random sample of the edge models), the latency does not increase significantly.

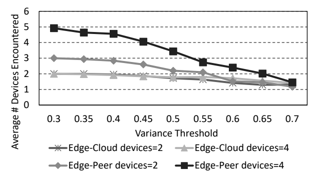
B. *Case Study: Edge Cloud Partitioning in Smart Industries*

The evolution of industry 4.0 [4] standard is bringing intelligent sensing and analytics into the industrial and manufacturing segment. Modern smart machines come with integrated sensors with built-in communication protocols to send data to either an attached computer, a base station or cloud. These features increase the cost of the machines significantly, making it extremely hard for small and medium scale entities to procure them. Moreover, a majority of the machines in operation, comprising much of the modern supply chain, are classical machines without any sensing or intelligence built into them. Without any smartness built into them, these classical machines often suffer from unforeseeable failures. Therefore, retrofitting such classical machines with smart sensors will help in preventing such failures and will allow taking predictive measures to increase production efficiency.

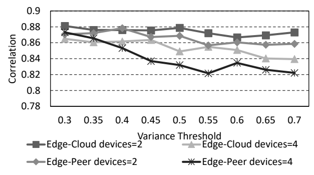
To understand the implications and benefits of retrofitting sensing into these classical machines, we conducted a case study on the data collected from a grinding machine which had three types of sensors – one power sensor and two accelerometers. It also incorporated the tool parameters like speed, feed



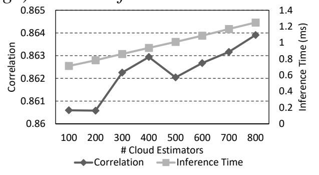
(a) Inference time vs threshold



(b) Communication through (average) number of devices vs Threshold



(c) Correlation vs threshold



(d) Accuracy (correlation) and inference time vs # cloud estimators

Fig. 4: Sensitivity study

and depth-of-cut, and measured the surface roughness of the grinding surface. The goal of our study was to correctly predict the surface roughness from sensor data. We divided the data into multiple chunks to emulate a multi machine setup (varying from 2 to 5). Since the data size is limited, with increasing numbers of machines, the training data-per-machine decreases, and hence gives us the opportunity to also study the impact of data availability. We implemented a random forest based regression model to predict the surface roughness value and tested our partitioning policies. The edge node is implemented on Raspberry Pi development boards and the cloud is emulated via a desktop class Intel corei9 10900k CPU (with 64GB DDR4 RAM). Figure 3 shows the accuracy and latency of different policies with different numbers of machines.

Key Observations:

1) If data is shared with (a third party) cloud, the accuracy of the model generated using all data from different machines is significantly higher than the models generated for the edge using their own data. Although, in case of 5 machines (see Figure 3) we see about a 5% accuracy gain with a 14.71% latency increment to perform the inference at cloud, this 5% increment has significant impact in practise. For example, for a typical grinding job that takes about 8.2 seconds [1], this 5% improvement impacts $\approx 46k$ parts per year per machine (working 8 hours/day).

2) For the most part, the accuracy of the privacy preserving model remains same (if not less) compared to the data sharing model (this reflects the fact that these machines might belong to different industries who were not willing to participate in the the data sharing process). Even with minimal accuracy improvement of 3.5% (see Figure 3, where the 5 machines belong to 5 different owners and not sharing data), one grinding machine can save up to $\approx 27.4k$ parts per year.

3) The latency difference between the data shared model and privacy preserved model is not so prominent due to the lower data volume. The models generated are simple, and hence does not have significant difference in execution time at the cloud.

Sensitivity Study: To better understand the relationship between the inference quality and the threshold, we run our model with different parameter settings, shown in Figure 4, over the cases of edge-peer and edge-cloud structures with device counts of two and four. Figure 4a shows that setting a high threshold value will reduce the average inference time in all four cases. This effect is more obvious in the edge-peer structure since it will send the data to its nearest peer and check the prediction quality until it reaches the cloud. Figure 4b also shows how network structure amplifies the effect of changing threshold value. Therefore, it is important to choose a threshold that optimizes the trade off between efficiency and accuracy. Similar to the data on inference time, the effect from the increasing threshold on prediction accuracy will be more pronounced when there are more devices available and the overall trend seen in Figure 4c is the inverse of that in Figure 4a. This makes it possible to find a sweet spot, given specific needs, in efficiency and accuracy trade offs. The number of estimators also needs to be customized for different datasets. Figure 4d shows that inference time will increase linearly with the number of estimators whereas it has a very small impact on correlation. Therefore, reducing the number of estimators while keeping the prediction quality over a certain boundary will optimize fine tuning the model.

IV. Conclusions

An efficient compute and data partitioning between edge and cloud, while preserving data privacy, is an important problems to address for both existing and future deployments. This work provides a practical solution to achieve latency-accuracy balanced partitions for random forest based inference tasks. We demonstrate the real-world applicability of our approach for two smart manufacturing deployments, and analyzed the accuracy-latency trade offs and their sensitivity to user-supplied thresholds. We believe that our solution can be easily deployed and be beneficial for random forest based distributed sensing-computing platforms.

Acknowledgements

We thank the reviewers for their helpful feedback. This research is supported by NSF grant #1822923 and Clean Energy Smart Manufacturing Innovation Institute award #136067.

References

[1] Automating the grinding process, 2013. https://www.sme.org/technologies/articles/2013/january/automating-the-grinding-process/.

[2] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems*, 1:374–388, 2019.

[3] Luis M. Candanedo, Véronique Feldheim, and Dominique Deramaix. Data driven prediction models of energy use of appliances in a low-energy house. *Energy and Buildings*, 140:81–97, 2017.

[4] Heiner Lasi, Peter Fettke, Hans-Georg Kemper, Thomas Feld, and Michael Hoffmann. Industry 4.0. *Business & information systems engineering*, 6(4):239–242, 2014.

[5] Yang Liu, Yingting Liu, Zhijie Liu, Yuxuan Liang, Chuishi Meng, Junbo Zhang, and Yu Zheng. Federated forest. *IEEE Transactions on Big Data*, 2020.

[6] Cyan Subhra Mishra, Jack Sampson, Mahmut Taylan Kandemir, and Vijaykrishnan Narayanan. Origin: Enabling on-device intelligence for human activity recognition using energy harvesting wireless sensor networks. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1414–1419. IEEE, 2021.

[7] Dazhong Wu, Connor Jennings, Janis Terpenny, Robert X Gao, and Soundar Kumara. A comparative study on machine learning algorithms for smart manufacturing: tool wear prediction using random forests. *Journal of Manufacturing Science and Engineering*, 139(7), 2017.