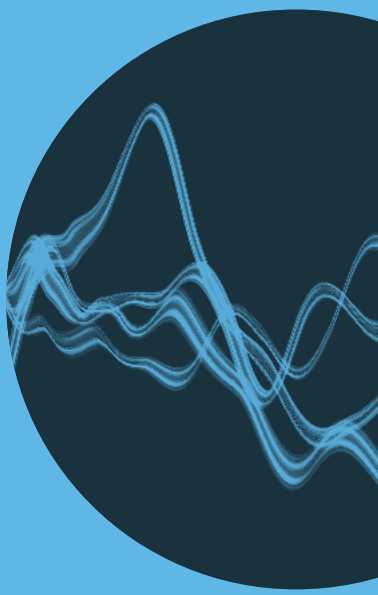
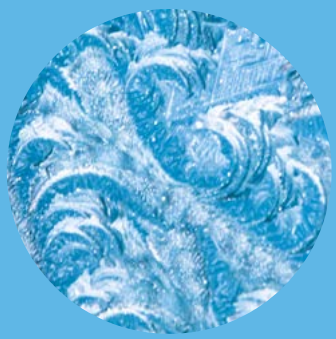
CHARTING THE FUTURE OF INNOVATION | #09·2019

ERICSSON TECHNOLOGY Review



PRIVACY-AWARE MACHINE LEARNING



ERICSSON

Privacy-aware machine learning

WITH LOW NETWORK FOOTPRINT

Federated learning makes it possible to train machine learning models without transferring potentially sensitive user data from devices or local deployments to a central server. As such, it addresses privacy concerns at the same time that it improves functionality. Depending on the complexity of the neural network, it can also dramatically reduce the amount of data needed while training a model.

KONSTANTINOS VANDIKAS, SELIM ICKIN, GAURAV DIXIT, MICHAEL BUISMAN, JONAS ÅKESON

Reliance on artificial intelligence (AI) and automation solutions is growing rapidly in the telecom industry as network complexity continues to expand. The machine learning (ML) models that many mobile network operators (MNOs) use to predict and solve issues before they affect user QoE are just one example.

An important aspect of the 5G evolution is the transformation of engineered networks into continuous learning networks in which self-adapting, scalable and intelligent agents can work independently to continuously improve quality and performance. These emerging "zero-touch networks" are, and will continue to be, heavily dependent on ML models.

The real-world performance of any ML model depends on the relevance of the data used to train it. Conventional ML models depend on the mass transfer of data from the devices or deployment sites to a central server to create a large, centralized dataset. Even in cases where the computation is decentralized, the training of conventional ML models still requires large, centralized datasets and misses out on using computational resources that may be available closer to where data is generated.

While conventional ML delivers a high level of accuracy, it can be problematic from a data security perspective, due to legal restrictions and/or privacy concerns. Further, the transfer of so much data requires significant network resources, which means that lack of bandwidth and data transfer costs can be an issue in some situations. Even in cases where all the required data is available, reliance on a centralized dataset for maintenance and retraining purposes can be costly and time consuming.

For both privacy and efficiency reasons, Ericsson believes that the zero-touch networks of the future must be able to learn without needing to transfer voluminous amounts of data, perform centralized computation and/or risk exposing sensitive information. Federated learning (FL), with its ability to do ML in a decentralized manner, is a promising approach.

To better understand the potential of FL in a telecom environment, we have tested it in a number of use cases, migrating the models from conventional, centralized ML to FL, using the accuracy of the original model as a baseline. Our research indicates that the usage of a simple neural network yields a significant reduction in network utilization, due to the sharp drop in the amount of data that needs to be shared.

As part of our work, we have also identified the properties necessary to create an FL framework that can achieve the high scalability and fault tolerance required to sustain several FL tasks in parallel. Another important aspect of our work in this area has been figuring out how to transfer an ML model that addresses a specific and common problem, pretrained within an FL mechanism on existing network nodes to newly joined network nodes, so that they too can benefit from what has been learned previously.

The concept of federated learning

The core concept behind FL is to train a centralized model on decentralized data that never leaves the local data center that generated it. Rather than transferring "the data to the computation," FL transfers "the computation to the data."[1]

In its simplest form, an FL framework makes use of neural networks, trained locally as close as possible to where the data is generated/collected. Such initial models are distributed to several data sources and trained in parallel. Once trained, the weights of all neurons of the neural network are transported to a central data center, where federated averaging takes place and a new model is produced and communicated back to all the remote neural networks that contributed to its creation.

Terms and abbreviations

AI – Artificial Intelligence | **AUC** – Area Under the Curve | **FL** – Federated Learning | **ML** – Machine Learning | **MNO** – Mobile Network Operator | **ROC** – Receiver Operating Characteristic

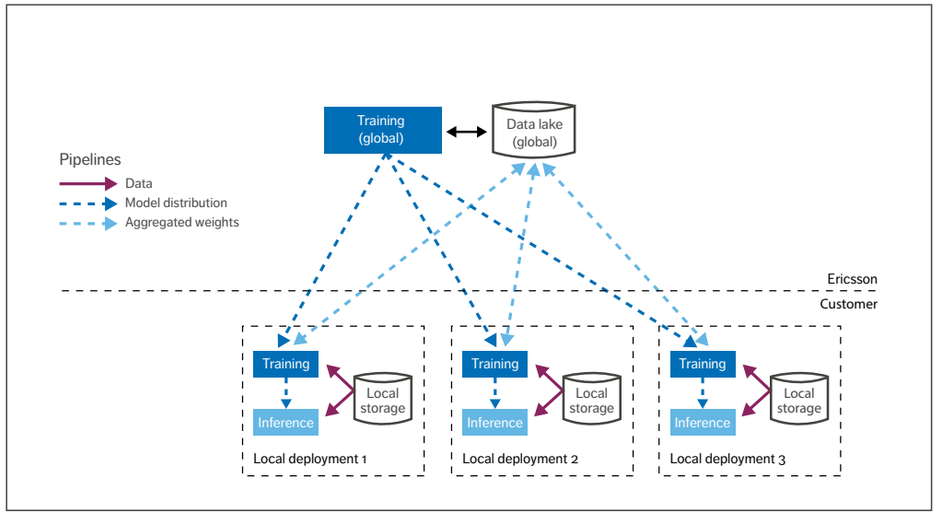


Figure 1 Overview of federated learning

Techniques such as secure aggregation [2] and differential privacy [3] can be applied to further ensure the privacy and anonymity of the data origin.

FL can be used to test and train not only on smartphones and tablets, but on all types of devices. This makes it possible for self-driving cars to train on aggregated real-world driver behavior, for example, and hospitals to improve diagnostics without breaching the privacy of their patients.

Figure 1 illustrates the basic architecture of an FL life cycle. The light blue dashed lines indicate that only the aggregated weights are sent to the global data lake, as opposed to the local data itself, as is the case in conventional ML models. As a result, FL makes it possible to achieve better utilization of resources, minimize data transfer and preserve the privacy of those whose information is being exchanged.

The main challenge with an FL approach is that the transition from training a conventional ML model using a centralized dataset to several smaller federated ones may introduce a bias that impacts the accuracy originally achieved by using a centralized dataset. The risk for this is greatest in less reliable and more ephemeral federations that span over to mobile devices.

It is reasonable to expect data centers used by MNOs to be significantly more reliable than devices in terms of data storage, computational resources and general availability. However, it is important to ensure high fault tolerance, as corresponding processes may still fail due to lack of resources, software bugs or other issues.

Federated learning framework design

Our FL framework design concept is cloud-native, built on a federation of Kubernetes-based data centers located in different parts of the world. We assume restricted access to allow for the execution of certain processes that are vital to FL.

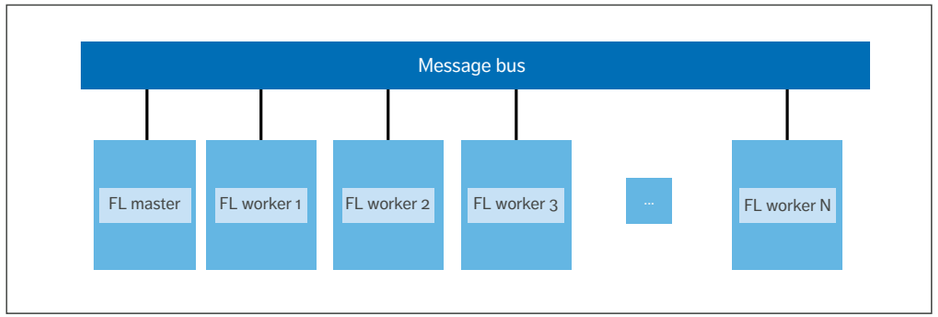


Figure 2 Basic design of an FL platform

Figure 2 illustrates the basic system design. A federation is treated as a task run-to-completion, enabling a single resource definition of all parameters of the federation that is later deployed to different cloud-native environments. The resource definition for the task deals both with variant and invariant parts of the federation.

The variant parts handle the characteristics of the FL model and its hyperparameters. The invariant parts handle the specifics of common components that can be reused by different FL tasks. Invariant parts include a message queue, the master of the FL task and the workers to be deployed and federated in different data centers.

Workers (processes running in local deployments) are tightly coupled with the underlying ML platform that is used to train the model, which is immutable during the federation. In our FL experiments, we selected TensorFlow to train the neural network, which is designed to be interchangeable with other ML platforms such as PyTorch. Communication between the master and the workers is protected using Transport Layer Security encryption with one-time generated public/private keys that are discarded as soon as an FL task is completed.

Invariant components can be reused by different FL tasks. FL tasks can run sequentially or in parallel depending on the availability of resources. Master and worker processes are implemented as stateless components. This design choice leads to a more robust framework, since it allows for an FL task to fail without affecting other ongoing FL tasks.

Fault tolerance

To reduce the complexity of the codebase for both the master of the FL task and the workers and to keep our implementation stateless, we chose a message bus to be the single point of failure in the design of our FL framework. This design choice is further motivated by the research into creating highly scalable and fault-tolerant message buses by combining leader-election techniques and/or by relying on persistent storage to maintain the state of the message queue in case of a failure [4].

The message exchange between the master of the FL task and the workers is implemented in the form of assigned tasks such as "compute new weights" and "average weights." Each task is pushed into the message queue and has a direct recipient. The recipient must acknowledge that it has received the task. If the acknowledgement is not made, the task remains in the queue. In case of a failure, messages remain in the message queue while Kubernetes restarts the failed process. Once the process reaches a running state again, the message queue retransmits any unacknowledged tasks.

Table A – Summary of scenario definitions

	Centralized	Isolated	Federated
Privacy preserved	No	Yes	Yes
Use of overall data	Yes	No	Yes
Data transfer cost	High	None	None
Weight transfer cost	None	None	Low

Table B – ROC AUC scores of workers throughout three scenarios

	Centralized median (std)	Isolated median (std)	Federated median (std)
Worker 1 (region 1)	0.91 (0.15)	0.89 (0.12)	0.95 (0.05)
Worker 2 (region 2)	0.92 (0.8)	0.93 (0.08)	0.93 (0.03)
Worker 3 (region 3)	0.95 (0.16)	0.95 (0.13)	0.97 (0.07)
Worker 4 (region 4)	0.97 (0.13)	0.97 (0.11)	0.96 (0.05)
Overall	0.93 (0.13)	0.93 (0.11)	0.95 (0.05)

Table C – Network footprint formulas for each training scenario

		Downlink consumption	Uplink consumption
Centralized ML	Master	$\sum_{i=0}^{N} n_i$	$N * \text{Model}_0$
	Worker ID	Model_0	n_i
Isolated ML	Master	0	0
	Worker ID	0	0
FL	Master	$N * R * \text{Model}_0$	$N * R * \text{Model}_0$
	Worker ID	$R * \text{Model}_0$	$R * \text{Model}_0$

i: worker ID
N: number of workers
R: number of rounds needed until accuracy convergence
Model_0: Size of ML model
n_i: size of dataset in worker ID

Table D – Network footprint

Centralized (MB)	Federated (MB)	FL message size (MB)	Rounds	Workers
2,000	19.2	0.26	15	4

Figure 3 Tables relating to the hardware fault prediction use case

Preventive maintenance use case

Hardware fault prediction is a typical ML use case for an MNO. In this case, the aim is to predict whether there will be a hardware fault at a radio unit within the next seven days based on data generated in the eight-week interval preceding the prediction time. The inputs to the ML model consist of more than 500 features that are aggregations of multiple performance management counters, fault management data such as alarms, weather data and the date/time since the hardware has been active in the field.

Three training scenarios

We performed the experiments in three scenarios – centralized ML, isolated ML and FL.

Centralized ML is the benchmark scenario. The datasets from all four worker nodes are transferred to one master node, and model training is performed there. The trained model is then transferred and deployed back to the four worker nodes for inference. In this scenario, all worker nodes use exactly the same pretrained ML model.

In the isolated ML scenario, no data is transferred from the worker nodes to a master node. Instead, each worker node trains on its own dataset and operates independently from the others.

In the FL scenario, the worker nodes train on their individual datasets and share the learned weights from the neural network model via the message queue. The saturation of the model accuracies is achieved after 15 rounds of the weight-sharing and weight-averaging procedure. In this way, the worker nodes can learn from each other without transferring their datasets.

The properties of each training scenario are summarized in *Figure 3*, Table A.

Accuracy results

Table B in Figure 3 presents the results in the form of median ROC AUC (receiver operating characteristic area under the curve) scores obtained through more than 100 independent experiments. The scores achieved in the FL scenario are similar to those achieved in the centralized and isolated ones, while the variance of the FL scores is significantly lower compared with the other two scenarios.

The results in Table B show that it is worker 1 (south) that benefits from FL. They also suggest that an isolated ML approach can be recommended in cases where the individual datasets have enough data for training. The only drawback is that because the isolated nodes never receive any information from other nodes, they will be more conservative in their response to changes in the data, with the risk of potentially higher blind spots in the individual datasets.

The impact of adding new workers

To facilitate the adding of new workers at a later time, information about the current round must be maintained in the message exchange between the master and the workers. When an FL task starts, all workers register to round ID 0, which triggers the master to initialize the random weights and broadcast the same distribution to all workers. All workers train in parallel and contribute to the same training round. As the rounds increase, the federated model's maturity increases until a saturation point is reached.

If the current round ID is greater than 0, the master is aware that the process of averaging of weights has taken place at least once, which means that the model is not at a random initial state. When a new worker joins the FL task, it sends its round ID as 0.

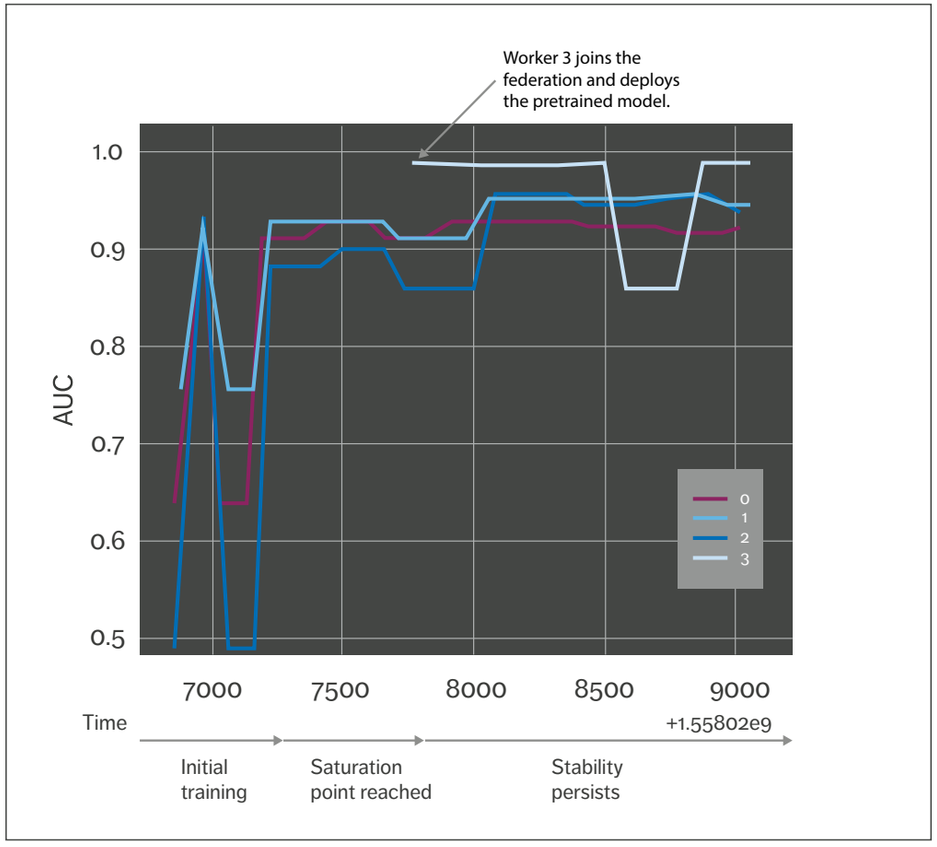


Figure 4 AUC scores before and after a new worker joins the FL task

The master, whose latest round ID is greater than 0, recognizes the worker as new and immediately shares the latest state of the model with the new worker after the first handshake.

Figure 4 illustrates how accuracy persists in the FL model when a new worker joins. In this example, three workers numbered 0, 1 and 2 contribute to the initial training phase of the model, receive the same randomized weight matrices from the master node, and train on the same model. As time passes, accuracy reaches a saturation point. Later, worker 3 joins the FL task. However, since worker 3 is at its first round, it is allowed to use the saturated model, but not allowed to contribute in the weight aggregation. Instead, its weights are discarded.

Network footprint comparison

Table C in Figure 3 presents a set of formulas that can be used to determine the network footprint for centralized ML, isolated ML and FL scenarios. When training an ML model in a conventional way (as in centralized ML), it is assumed that the number of message exchanges is equal to the number of regions where the compressed version of the dataset is transferred – that is, the number of local deployments.

In FL, on the other hand, there are multiple message exchanges that consist of rounds, which refers to the number of training phases required until the accuracy of the model converges. Message size in FL is determined by the serialized vector that contains the neural weights, which is directly related to the size of the neural network.

Table D in Figure 3 presents the network footprint results for the preventive maintenance use case, showing that the FL approach yielded a one order of magnitude reduction in data volume compared with the centralized approach – a drop from 2,000MB to 19.2MB. This dramatic reduction can be attributed to the simplicity of the neural network and the substantially smaller amount of data that needs to be transferred in the FL scenario. In the long term, this significantly smaller network footprint will enable the creation of a more complex neural network with the ability to detect more complex patterns in the dataset.

Conclusion

While conventional machine learning (ML) models provide many benefits to mobile network operators, particularly in terms of ensuring consistent QoE, the large data transfer that they require results in a substantial network footprint and can lead to privacy issues. By bringing "the computation to the data" instead of transferring "the data to the computation," federated learning (FL) makes it possible to overcome those challenges by training a centralized model on decentralized data.

Ericsson's research in this area has demonstrated that it is possible to migrate from a conventional ML model (trained using a completely centralized dataset) to a federated one, significantly reducing data transfer and protecting privacy while achieving on-par accuracy. In light of our findings, we believe that FL has an important role to play in the ongoing automation of the telecom sector and in the transition to the zero-touch networks of the future.

References

1. **Communication-Efficient Learning of Deep Networks from Decentralized Data, February 28, 2017, H. Brendan McMahan et al., available at:** *https://arxiv.org/pdf/1602.05629.pdf*
2. **Google, Practical Secure Aggregation for Privacy-Preserving Machine Learning, 2017, K. Bonawitz et al., available at:** *https://ai.google/research/pubs/pub47246/*
3. **Deep Learning with Differential Privacy, October 25, 2016, M. Abadi et al., available at:** *https://arxiv.org/pdf/1607.00133.pdf*
4. **Distributed Systems: Principles and Paradigms, 2002, A. Tanenbaum et al., available at:** *https://www.amazon.com/Distributed-Systems-Principles-Andrew-Tanenbaum/dp/0130888931*

Further reading

» **Privacy Preserving QoE Modeling using Collaborative Learning, October 21, 2019, Selim Ickin, Konstantinos Vandikas, and Markus Fiedler, in the 4th Internet-QoE Workshop: QoE-based Analysis and Management of Data Communication Networks (Internet-QoE'19), ACM, New York, NY, USA, available at:** *https://doi.org/10.1145/3349611.3355548*

THE AUTHORS

Konstantinos Vandikas

◆ is a principal researcher at Ericsson Research whose work focuses on the intersection between distributed systems and AI. His background is in distributed systems and service-oriented architectures. He has been with Ericsson Research since 2007, actively evolving research concepts from inception to commercialization. Vandikas has 23 granted patents and over 60 patent applications. He has authored or coauthored more than 20 scientific publications and has participated in technical committees at several conferences in the areas of cloud computing, the Internet of Things and AI. He holds a Ph.D. in computer science from RWTH Aachen University, Germany.

Selim Ickin

◆ joined Ericsson Research in 2014 and is currently a senior researcher in the AI department in Sweden. His work has been mostly around building ML prototypes in diverse domains such as to improve network-based video streaming performance, to reduce subscriber churn rate for a video service provider and to reduce network operation cost. He holds a B.Sc. in electrical and

electronics engineering from Bilkent University in Ankara, Turkey, as well as an M.Sc. and a Ph.D. in computing from Blekinge Institute of Technology in Sweden. He has authored or coauthored more than 20 publications since 2010. He also has patents in the area of ML within the scope of radio network applications.

Gaurav Dixit

◆ joined Ericsson in 2012. He currently heads the Automation and AI Development function for Business Area Managed Services. In earlier roles he was a member of the team that set up the cloud product

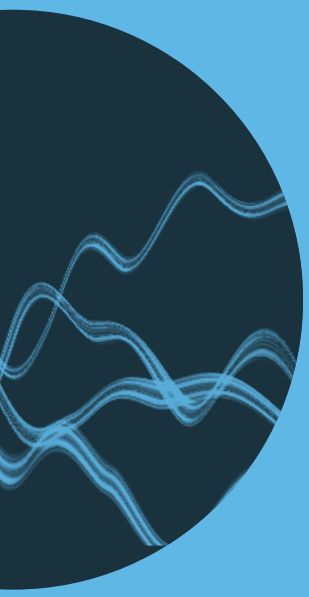
business within Ericsson. He holds an MBA from the Indian Institute of Management in Lucknow, India, and the Università Bocconi in Milan, Italy, as well as a B.Tech. in electronics and communication engineering from the National Institute of Technology in Jalandhar, India.

Michael Buisman

◆ is a strategic systems director at Business Area Managed Services whose work focuses on ML and AI. He joined Ericsson in 2007 and has more than 20 years of experience of delivering new innovations in the telecom industry that drive the transition to a digital world. For the past two years, Buisman and his team have been developing a managed services ML/AI solution that is now being deployed to several customers globally. Buisman holds a BA from the University of Portsmouth in the UK and an MBA from St. Joseph's University in Philadelphia in the US.

Jonas Åkeson

◆ joined Ericsson in 2005. In his current role, he drives the implementation of AI and automation in the three areas that integrate Ericsson's Managed Services business. He holds an M.Sc. in engineering from Linköping Institute of Technology, Sweden, and a higher education diploma in business economics from Stockholm University, Sweden.



ISSN 0014-0171
284 23-3333 | Uen

Ericsson
SE-164 83 Stockholm, Sweden
Phone: +46 10 719 0000