Collaborative Learning on the Edges: A Case Study on Connected Vehicles

Sidi Lu, Yongtao Yao, Weisong Shi
{lu.sidi, yongtao.yao, weisong}@wayne.edu
Wayne State University

Abstract

The wide deployment of 4G/5G has enabled connected vehicles as a perfect edge computing platform for a plethora of new services which are impossible before, such as remote real-time diagnostics and advanced driver assistance. In this work, we propose CLONE, a collaborative learning setting on the edges based on the real-world dataset collected from a large electric vehicle (EV) company. Our approach is built on top of the federated learning algorithm and long short-term memory networks, and it demonstrates the effectiveness of driver personalization, privacy serving, latency reduction (asynchronous execution), and security protection. We choose the failure of EV battery and associated accessories as our case study to show how the CLONE solution can accurately predict failures to ensure sustainable and reliable driving in a collaborative fashion.

1 Introduction

The proliferation of edge computing technologies is strongly stimulating the adoption of machine learning methods on vehicles so that they can provide a variety of intelligent onboard services. On the one hand, current vehicles, such as connected and autonomous vehicles (CAVs), can generate over 11 TB of privacy-sensitive data per day [43]. Such big data amount brings not only opportunities but also challenges to domain researchers - practices have proved that larger training data set can achieve more remarkable results [40]; however, training a big model often requires excessive computation and memory resources, which hinders the application of machine learning algorithms on the resource-constrained edge devices [47].

On the other hand, electric vehicles (EVs) have received significant attention as an important efficient and sustainable transportation system. As a key component of EVs, the battery system largely determines the safety and durability of EVs [2, 50]. Due to the aging process or abuse maneuvers, various faults may occur at each constituent cell or associated accessories. It is essential to develop early failure detection techniques for EV battery and associated accessories to ensure availability and safety of EVs through anticipated replacements.

Besides, although EIC (electric, instrumentation and computer control system) data (such as voltage and current) of EVs are able to show the symptoms of an imminent failure of EV battery and associated accessories, they are hard to tell the reason why the battery and associated accessories failed. Driver behavior metrics such as speed, acceleration, and steering reflect the usage of an EV. We believe such usage is one of the main root causes of failures.

Hence, in this paper, we choose the failure of EV battery and associated accessories as our case study and seek to answer the following vital questions: *"How to construct a personalized model by continuously tuning parameters on connected vehicles?", "What is the influence of the driver behavior metrics on the failure prediction of EVs?".*

Our main contributions include:

- To the best of our knowledge, this is the first work to predict an imminent failure of EV battery and associated accessories based on the real-world EV dataset which involves EIC attributes and driver behavior metrics.
- Our analysis reveals that adding driver behavior metrics can improve the prediction accuracy of EV failures.
- We train random forest (RF), gradient boosting decision tree (GBDT), and long short-term memory networks (LSTMs) to predict failures, and we find LSTMs outperform other methods based on our dataset.
- We propose CLONE, a collaborative learning setting on the edges for connected vehicles, and it can reduce training time significantly without sacrificing prediction accuracy.

2 Data Description

This study presents the analysis of EV health characteristics based on the data measured at and collected from a large EV company. We analyze three different models of EVs, and the corresponding data is reported and collected every 10 milliseconds during the whole 6-hour collection period.

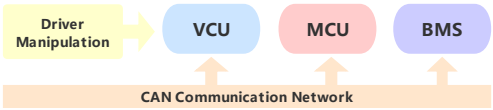

Figure 1: Three Core Control Systems of EVs.

In general, our data set is collected from the core control systems of EVs, which includes the vehicle control unit (VCU) [34], motor control unit (MCU) [30], and battery management system (BMS) [7]. BMS [7] is responsible for the battery maintenance and state estimation. The VCU [34], as a key component of the whole EV, sends orders to other modules based on the driver manipulation (such as gear signal, accelerator pedal signal, and vehicle mode) via CAN communication network [41]. MCU [30] controls the wheel motor locally according to the commands from VCU. Figure 1 shows the structure diagram of the core control systems. 42 features listed in Table 1 and Table 2 were analyzed.

Table 1: Selected EIC features.

Voltage	Temperature	Power & Energy
BMS_BattVolt	InCar_Temp	BMS_BattSOC
BMS_CellVoltMax	Environment_Temp	BMS_MaxChgPwrCont
BMS_CellVoltMax_Num	BMS_BattTempAvg	BMS_MaxChgPwrPeak
BMS_CellVoltMin	BMS_Inlet_WaterTemp	BMS_MaxDchgPwrCont
BMS_CellVoltMin_Num	BMS_Outlet_WaterTemp	BMS_MaxDchgPwrPeak
MCUF_Volt	BMS_MaxTemp	VCU_Batt_Comp_Pwr
MCUR_Volt	BMS_MinTemp	VCU_Batt_PTC_Pwr
Current	BMS_TempMaxNum	**Error Info**
BMS_BattCurr	BMS_TempMinNum	BMS_BatterySysFaultLevel
MCUF_Curr	MCUF_Temp	BMS_Low_SOC
MCUR_Curr	MCUR_Temp	VCU_PTC_ErrSta

Table 2: Selected driver behavior metrics.

Driver Behavior	VehicleSpeed	Acceleration	Steering
	YawRate	WheelSpeedFL	WheelSpeedFR
	WheelSpeedRL	WheelSpeedRR	Emergency_Stop
	AccPedalPosition	BrakePedalPosition	

Driver behavior metrics are collected from VCU and sensors.

More specifically, we analyze EV data in the two aspects: (1) EIC attributes, and (2) driver behavior metrics. Here, EIC refers to electric, instrumentation, and computer control system [29]. It includes battery features collected from BMS (most commonly used for EV battery durability analysis by other studies [2, 16, 39, 44, 51]) and the data reported from other control systems. Most of the selected features can be understood intuitively; hence, we choose some vague features to give our explanations. "MCUF" represent the MCU for the front wheels, and "MCUR" means the MCU for the rear wheels. Positive Temperature Coefficient heater (PTC) is the heating unit of the battery in EVs. Besides, state of charge (SOC) is the indicator of battery left capacity, and "Comp" is an acronym of the compressor.

3 Collaborative Learning on Edges (CLONE)

The models learned in this paper are implemented in Python, using tensorflow 1.5.0 [1], keras 2.1.5 [21], and scikit-learn libraries [38] for model building. We use 5-fold cross-validation method [27, 42] to evaluate the proposed prediction approach. Note that there is a trade-off between long prediction horizon and the sampling frequency, with the constraints of computing resources. After conducting a series of sensitivity study, we choose 15 seconds as our prediction horizon so that it can predict failures for the next 1,500 data points.

3.1 Stand-alone Learning

Before employing CLONE, we first combine the whole real-world dataset of three EVs to train different machine learning models on the Intel FogNode (the hardware information is shown in Table 5). The goal is to find a suitable algorithm to predict failures, and answer the question of "What is the influence of the driver behavior metrics on EV failure prediction?"

We tackle the failure prediction problem using random forest (RF) [31], gradient boosted decision tree (GBDT) [17, 52], and long short-term memory networks (LSTMs) [13, 23] since they have become highly successful learning models for both classification and regression problems.

To show the impact of driver behavior metrics on the failure prediction, we conduct experiments on two experimental groups. Our first step is to combine all selected EIC attributes and driver behavior metrics to train models using RF, GBDT, and LSTMs methods, and we label this group as ED Group. Then, we exclude all driver behavior metrics but keep EIC attributes, and we denote it as E Group. Table 3 shows the input features for ED Group and E Group.

Table 3: Input features for two experimental groups.

	EIC attributes	Driver Behavior Metrics
ED Group	31 attributes	11 metrics
E Group	31 attributes	NONE

Table 4: Evaluation results.

		Precision	Recall	Accuracy	F-measure
ED Group	**RF**	0.7492	0.7814	0.7833	0.7649
	GBDT	0.7905	0.8500	0.8234	0.8192
	LSTM	0.9420	0.9500	0.9430	0.9460
	Average	0.8272	0.8605	0.8499	0.8434
E Group	**RF**	0.6615	0.6900	0.7008	0.6755
	GBDT	0.6975	0.7500	0.7294	0.7228
	LSTM	0.8924	0.9000	0.8738	0.8962
	Average	0.7505	0.7800	0.7680	0.7648

Table 4 presents the average evaluation scores of ED and E Group. Based on our experimental results, we have the following observations:

- Excluding driver behavior metrics results in around 8% reduction in the average F-measure.
- LSTMs outperform RF and GBDT in both two groups based on our dataset.

3.2 CLONE Design

By observing experiment results of stand-alone learning, we can see that driver behavior metrics has non-negligible impacts on the failure prediction, and employing LSTMs can

achieve better results. Therefore, we aim to deploy LSTMs-based collaborative learning approaches on the edges based on EIC attributes and driver behavior metrics. We term our approach CLONE, which is the solution of the problems - "How to construct a personalized model on connected vehicles?".

3.2.1 Model Description

The learning tasks of CLONE is solved by a group of distributed participating vehicles (edge nodes) which are coordinated by a Parameter EdgeServer. Each vehicle has its local training dataset which is never uploaded to the Parameter EdgeServer or transferred to the cloud. Instead, each vehicle is responsible for continuously performing training and inference locally based on its private data. When a vehicle finishes one epoch [19], which refers to the number of iteration related with the input dataset during training, it will push the value of current parameters to the Parameter EdgeServer, where the parameter values are aggregated by computing the weighted average value. Then, each vehicle can immediately pull the updated parameter values from the Parameter EdgeServer, and set the updated parameters as their current parameters to start the next epoch. The above steps will be repeated as necessary. Figure 2 shows the basic framework of CLONE.

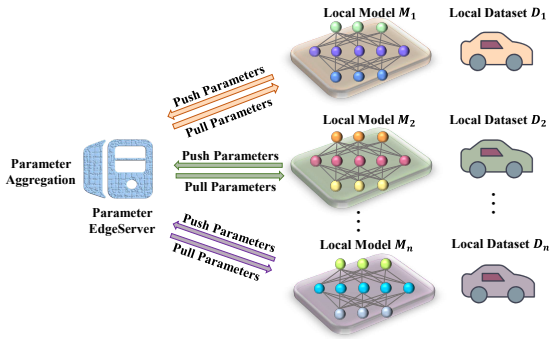

Figure 2: The framework of CLONE. *In CLONE, each vehicle trains the neural network model locally based on its private data. Then, the value of current parameters from each vehicle is uploaded to the Parameter EdgeServer, where those parameters are aggregated and sent back to vehicles.*

Note that when a new vehicle joins in, it will pull the current aggregated parameters from the Parameter EdgeServer first, and set them as the initial parameters for the first round of training, which speeds up the training process of unseen vehicles. Besides, since it is asynchronous communication, for each vehicle, there is no need to stop and wait for other vehicles to complete an epoch, which greatly reduces the latency. To illustrate the aggregation protocol of this work, we need to introduce the loss function first, which is defined as follows:

$$Loss = \sum_i \left[\hat{y}_{(i)} * log(y_{(i)}) + (1-\hat{y}_{(i)}) * log(1-y_{(i)})\right]$$

Here, $\hat{y}_i$ is the predicted output of the machine learning model, and the scalar y_i is the desired output of the model for each data sample i. We then define the formula to aggregate and update parameters as:

$$\begin{cases} P(p) \leftarrow \frac{Loss(v)}{Loss(p)+Loss(v)} P(p) + \frac{Loss(p)}{Loss(p)+Loss(v)} P(v) \\ Loss(p) \leftarrow Loss(v) \end{cases}$$

Where P represents the value of a parameter, and $Loss$ stands for the value of the loss function. Besides, p refers to the Parameter EdgeServer, and v represents a specific vehicle. For the more accurate vehicle (lower value of loss function), we assign a higher weight to its parameter.

3.2.2 Hardware Setup

To build heterogeneous hardware cluster representing different models of EVs, we adopt two different types of hardware - Intel FogNode and Jetson TX2, with different CPUs, operating systems and so on (shown in Table 5). More specifically, we choose one Intel FogNode as the Parameter EdgeSever, and we treat the other two Intel FodeNodes and one Jetson TX2 as the edge nodes (vehicles) to continuously "learn" latent patterns.

Table 5: Hardware setup for CLONE.

	Intel FogNode	**Jetson TX2**
CPU	Intel Xeon E3-1275 v5	ARMv8 + NVIDIA Pascal GPU
Frequency	3.6 GHz	2 GHz
Cores	4	6
Memory	32 GB	8 GB
OS	Linux 4.13.0-32-generic	Linux 4.4.38-tegra

3.2.3 Model Setup

In Section 3.1, we trained an accurate LSTMs model with 4 layers on the front and followed by a fully connected layer (dense layer). Now, we aim to deploy a collaborative LSTMs with the same number of layers on the edges, i.e., with the same hyperparameters. We first distribute our whole dataset to three edge nodes so that each edge node (vehicle) has its locally private dataset.

Table 6: Model parameters.

Layers	**Variables**	**Shape**
First Layer (lstm_1)	lstm_1/kernel	(16, 400)
	lstm_1/recurrent_kernel	(100, 400)
	lstm_1/bias	(400,1)
Second Layer (lstm_2)	lstm_2/kernel	(100, 400)
	lstm_2/recurrent_kernel	(100, 400)
	lstm_2/bias	(400,1)
Last Layer (dense_1)	dense_1/kernel	(100, 24)
	dense_1/bias	(24,1)

Table 6 shows the parameter distribution of the LSTMs model on the first two LSTMs layers (marked as lstm_1 and lstm_2) and the last fully connected layer (labelled as dense_1). The "kernel" and "recurrent_kernel" are the parameter vectors, and the last column represents the shape (size) of the parameters for each vector. For example, (16, 400) indicates that there are 16 $\times$ 400 of parameters. Our whole

network contains up to 297,700 parameters, including the weights and the biases. Weight can reflect the strength of the connection between input and output. Bias shows how far off the predictions are from the real values.

3.2.4 Throughput

Figure 3 shows the I/O throughput per second at the Parameter EdgeServer when the three edge nodes are working at the same time. It can be seen that the peak of the data throughput is relatively stable, and the peak appears intermittently. Besides, the maximum I/O throughput for push and pull process is around 750 KB/s and 250 KB/s respectively, which indicates that there is no big pressure on the network throughput.

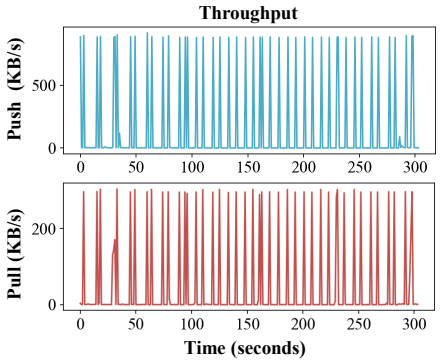

Figure 3: I/O throughput per second.

Figure 3 also proves that the push process is usually much slower than the pull process which was concluded by the work of [28]. This observation shows the importance to investigate methods that can reduce the communication latency of push process in the future work.

4 Evaluation

In this section, we present the experimental results of CLONE, and compare it with the algorithm performance of stand-alone learning in two aspects - (1) training time, and (2) evaluation scores including precision, recall, accuracy, and F-measure.

To have a clear comparison, we conduct experiments on three experimental groups. The first group is the stand-alone learning, and we set the epoch of stand-alone learning equal to 210. The second group is CLONE with the epoch of 70 for each edge node, and we label it as CLONE1. Since there are three edge nodes in CLONE1, the equivalent number of iterations in total is also 210 (70×3). As to the third group (CLONE2), the epoch is 100 for a single edge node, which results in 300 (100×3) of total iterations.

4.1 Training Time Comparison

We first profile and compare how the training time is spent on the three experimental groups, which is shown in Table 7.

Table 7: Training time (seconds).

	Intel FogNode1	Intel FogNode2	Jetson TX2
Stand-alone learning(epoch=210)	1183	1573	1497
CLONE1(epoch=70×3)	657	734	765
CLONE2(epoch=100×3)	928	1036	1158

For the stand-alone learning, the used training time varies with different edge devices - it takes 1183s and 1573s on two Intel FogNodes (different working states) respectively, while taking 1497s to execute the training task on Jetson TX2. As to CLONE1, the training time of each edge node is much lower than the training time of the single edge node of stand-alone learning. Since there are three edge nodes in CLONE1, the training time of CLONE1 should be one-third of stand-alone learning theoretically. However, due to the inevitable delay of the parameter transmissions, the training time of CLONE1 is greater than one-third of stand-alone learning. We then increase the epoch value from 70×3 to 100×3 (CLONE2), we can see the required training time is longer than CLONE1 as it has a larger number of iteration related with the input dataset during training, but it still less than stand-alone learning which has a lower epoch value. *Note that with the participation of more edge nodes and larger size of the input dataset, the advantages of CLONE in training time reduction will be more obvious.*

4.2 Evaluation Scores Comparison

We then calculate the average evaluation scores for each group, which is shown in Figure 4. Compared stand-alone learning and CLONE1, we can see that the overall evaluation scores of CLONE1 are lower than stand-alone learning. This may be caused by the fact that - in stand-alone learning, the prediction accuracy will be improved with the increasing number of iterations passing the full dataset through the current model. However, in CLONE1, due to the hardware difference, powerful edge nodes may train the model with high accuracy prior to other edge nodes. When the parameters of the poor training results are uploaded to the Parameter EdgeServer, the global accuracy of CLONE1 will be influenced. This may explain the performance gap between stand-alone learning and CLONE1 whose total epoch values are the same.

However, when we further increase the value of epoch (CLONE2), it can achieve high evaluation scores as stand-alone learning. Note that, by observing Table 7, the training time of CLONE2 is much lower than stand-alone learning, even though CLONE2 has higher epoch.

5 Discussion

Compared with stand-alone learning, CLONE can reduce model training time without sacrificing algorithm performance. With more edge nodes involved, the advantages of CLONE in training time reduction will be more obvious. Besides, compared with the collaborative cloud-edge approach, the main advantages of CLONE is to speed up the analysis

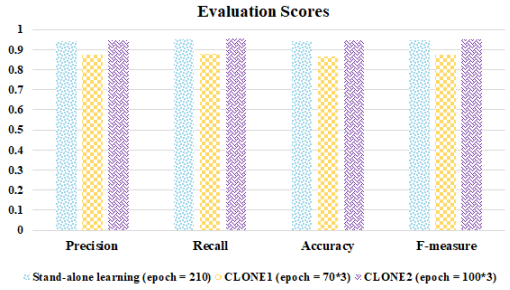

Figure 4: Algorithm performance.

tasks and protect user privacy better as it does not need to transfer any portion of the big and sensitive dataset via the network.

5.1 Possible Improvements

There are some possible improvements for CLONE. We list three of them for the discussion.

- **Bandwidth demand** - As the increasing number of edge nodes or the participation of larger neural networks, the communication of CLONE may be limited by bandwidth. In this context, we can use the Parameter EdgeServer group. In the group, Parameter EdgeServers can communicate with each other. Each Parameter EdgeServer is only responsible for a portion of parameters, and they work together to maintain globally shared parameters and their updates.
- **Aggregation protocol** - It is essential to find a suitable aggregation rule for the Parameter EdgeServer to aggregate parameters, which requires excessive experiments based on the specific experimental conditions.
- **Push latency** - Pushing parameters to the Parameter EdgeServer is usually much slower than pulling parameters. Hence, it is essential to investigate methods to reduce uplink latency (possible solutions include structured updates and sketched updates [28]).

5.2 Potential Use Cases

Besides, there are a variety of other meaningful use cases that CLONE can help, particularly for two types of scenarios:

- Real-time applications which requires developing suitable machine learning algorithms on the resource-constrained edges.
- Due to the privacy or/and the large network bandwidth constraints, the training dataset cannot be moved away from its source.

6 Related Work

Although machine learning algorithms are widely deployed, it is difficult to deploy them on the resource-constrained devices. In this context, model compression technologies [9, 11, 12, 22, 33, 45] and lightweight machine learning algorithms [8, 24, 25] have been proposed, but they can not guarantee to solve the problem completely when the training data and machine learning models are particularly large. Another popular choice to address this limitations is employing distributed data flow systems such as MapReduce [10], Spark [54], Naiad [36], and XGBoost [6]. They are able to robustly scale with the increasing dataset size, but when training complex neural networks tasks, the data flow systems fail to scale as they are inefficient at executing iterative workloads [3, 55].

This restriction sparked the development of distributed machine learning (DML) algorithms [5, 14, 15, 32, 53]. Later on, federated learning (FL), a novel DML, was proposed by Google researchers [4, 18, 28, 35, 49]. The main difference between conventional DML and FL is that, in FL, data is collected at the edges directly and stored persistently; thus, the data distribution at different edge nodes are usually not independent and identically distributed (non-i.i.d) [48]. Our work is inspired by FL, and the advantages of CLONE on the vehicles are shown in Table 8.

Table 8: Advantages of CLONE.

Advantages	Description	Outperform
Driver Personalization	Each vehicle trains the neural network model locally based on its private data; local models will be updated according to the dynamic changes of the local dataset.	DML, Cloud-based method
Privacy-Preserving	The training data can always be kept in its original location.	DML, Cloud-based method
Asynchronous Execution	There is no need to stop and wait for other vehicles to perform an iteration; solve the inefficient communication problem of bulk synchronous execution.	DML
Latency Reduction	Analyze vehicle data onboard; vehicles just need to push the parameter value to the Parameter EdgeServer rather than the whole data set.	DML, Cloud-based method, Cloud-edge method
Security Protection	Reduce security risks by limiting the attack surface to only the edges, instead of the edges and the cloud.	Cloud-based method, Cloud-edge method

Different from the collaborative cloud-edge method that a few papers proposed [20, 26, 48], CLONE has three main strengths - (1) reduces power consumption by eliminating the use of central data centers, (2) speeds up the analysis and modeling tasks as it always analyze real-time data onboard and just need to communicate with the Parameter EdgeServer about the current parameters [37, 46], and (3) reduces security risk by limiting the attack surface to only the edges.

7 Conclusion and Future Work

In this paper, we conduct a field study of EVs based on a real-world dataset collected from a large EV company. We discover that driver behavior metrics are potentially good indicators of the failures of EV battery and associated accessories. Besides, we propose CLONE, collaborative learning setting on the edges for connected vehicles, which can reduce model training time significantly. In the future, we plan to explore more advanced neural networks, enlarge the applying scope, and find a more suitable aggregation protocol for CLONE.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, volume 16, pages 265–283, 2016.

[2] Li Bao, Lingling Fan, and Zhixin Miao. Real-time simulation of electric vehicle battery charging systems. In *2018 North American Power Symposium (NAPS)*, pages 1–6. IEEE, 2018.

[3] Christoph Boden, Tilmann Rabl, and Volker Markl. Distributed machine learning-but at what cost. In *Machine Learning Systems Workshop at the 2017 Conference on Neural Information Processing Systems*, 2017.

[4] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konecny, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

[5] Yingyi Bu, Bill Howe, Magdalena Balazinska, and Michael D Ernst. Haloop: efficient iterative data processing on large clusters. *Proceedings of the VLDB Endowment*, 3(1-2):285–296, 2010.

[6] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[7] Ka Wai Eric Cheng, BP Divakar, Hongjie Wu, Kai Ding, and Ho Fai Ho. Battery-management system (bms) and soc development for electrical vehicles. *IEEE transactions on vehicular technology*, 60(1):76–88, 2011.

[8] François Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint*, pages 1610–02357, 2017.

[9] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre Binaryconnect David. Training deep neural networks with binary weights during propagations. arxiv preprint. *arXiv preprint arXiv:1511.00363*, 2015.

[10] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[11] Misha Denil, Babak Shakibi, Laurent Dinh, Nando De Freitas, et al. Predicting parameters in deep learning. In *Advances in neural information processing systems*, pages 2148–2156, 2013.

[12] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*, pages 1269–1277, 2014.

[13] Fernando Dione dos Santos Lima, Gabriel Maia Rocha Amaral, Lucas Goncalves de Moura Leite, João Paulo Pordeus Gomes, and Javam de Castro Machado. Predicting failures in hard drives with lstm networks. In *Proceedings of the 2017 Brazilian Conference on Intelligent Systems (BRACIS)*, pages 222–227. IEEE, 2017.

[14] Jaliya Ekanayake, Hui Li, Bingjing Zhang, Thilina Gunarathne, Seung-Hee Bae, Judy Qiu, and Geoffrey Fox. Twister: a runtime for iterative mapreduce. In *Proceedings of the 19th ACM international symposium on high performance distributed computing*, pages 810–818. ACM, 2010.

[15] Stephan Ewen, Kostas Tzoumas, Moritz Kaufmann, and Volker Markl. Spinning fast iterative data flows. *Proceedings of the VLDB Endowment*, 5(11):1268–1279, 2012.

[16] Abbas Fotouhi, Daniel J Auger, Karsten Propp, Stefano Longo, and Mark Wild. A review on electric vehicle battery modelling: From lithium-ion toward lithium–sulphur. *Renewable and Sustainable Energy Reviews*, 56:1008–1021, 2016.

[17] Jerome H Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[18] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

[19] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

[20] Philipp M Grulich and Faisal Nawab. Collaborative edge and cloud neural networks for real-time video processing. *Proceedings of the VLDB Endowment*, 11(12):2046–2049, 2018.

[21] Antonio Gulli and Sujit Pal. *Deep Learning with Keras*. Packt Publishing Ltd, 2017.

[22] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[23] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[24] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[25] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[26] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *Acm Sigplan Notices*, 52(4):615–629, 2017.

[27] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence (IJCAI)*, volume 14, pages 1137–1145, 1995.

[28] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[29] Bo Li, Weina Wang, Long Jia, Dafang Wang, and Anran Kong. Study on hil system of electric vehicle controller based on ni. In *IOP Conference Series: Materials Science and Engineering*, volume 382, page 052033. IOP Publishing, 2018.

[30] Hong-Peng Li and Yan-wen Li. The research of electric vehicle's mcu system based on iso26262. In *2017 2nd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*, pages 336–340. IEEE, 2017.

[31] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomForest. *R news*, 2(3):18–22, 2002.

[32] Yucheng Low, Joseph E Gonzalez, Aapo Kyrola, Danny Bickson, Carlos E Guestrin, and Joseph Hellerstein. Graphlab: A new framework for parallel machine learning. *arXiv preprint arXiv:1408.2041*, 2014.

[33] Ping Luo, Zhenyao Zhu, Ziwei Liu, Xiaogang Wang, Xiaoou Tang, et al. Face model compression by distilling knowledge from neurons. In *AAAI*, pages 3560–3566, 2016.

[34] Yan Ma, Kangkang Zhang, Jing Gu, Jianqiu Li, and Dongbin Lu. Design of the control system for a four-wheel driven micro electric vehicle. In *2009 IEEE Vehicle Power and Propulsion Conference*, pages 1813–1816. IEEE, 2009.

[35] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

[36] Derek G Murray, Frank McSherry, Rebecca Isaacs, Michael Isard, Paul Barham, and Martín Abadi. Naiad: a timely dataflow system. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, pages 439–455. ACM, 2013.

[37] Donghyun Park, Seulgi Kim, Yelin An, and Jae-Yoon Jung. Lired: A light-weight real-time fault detection system for edge computing using lstm recurrent neural networks. *Sensors*, 18(7):2110, 2018.

[38] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[39] E Peled, D Golodnitsky, H Mazor, M Goor, and S Avshalomov. Parameter analysis of a practical lithium- and sodium-air electric vehicle battery. *Journal of Power Sources*, 196(16):6835–6840, 2011.

[40] Junfei Qiu, Qihui Wu, Guoru Ding, Yuhua Xu, and Shuo Feng. A survey of machine learning for big data processing. *EURASIP Journal on Advances in Signal Processing*, 2016(1):67, 2016.

[41] Li Ran, Wu Junfeng, Wang Haiying, and Li Gechen. Design method of can bus network communication structure for electric vehicle. In *International Forum on Strategic Technology 2010*, pages 326–329. IEEE, 2010.

[42] Juan D Rodriguez, Aritz Perez, and Jose A Lozano. Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE transactions on pattern analysis and machine intelligence (TPAMI)*, 32(3):569–575, 2010.

[43] Tiffiny Rossi. Autonomous and adas test cars produce over 11 tb of data per day (article). October 10, 2018.

[44] Kaveh Sarrafan, Kashem M Muttaqi, and Danny Sutanto. Real-time state-of-charge tracking system using mixed estimation algorithm for electric vehicle battery system. In *2018 IEEE Industry Applications Society Annual Meeting (IAS)*, pages 1–8. IEEE, 2018.

[45] Bharat Bhusan Sau and Vineeth N Balasubramanian. Deep model compression: Distilling knowledge from noisy teachers. *arXiv preprint arXiv:1610.09650*, 2016.

[46] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[47] Gregor Ulm, Emil Gustavsson, and Mats Jirstrand. Oodida: On-board/off-board distributed data analytics for connected vehicles. *arXiv preprint arXiv:1902.00319*, 2019.

[48] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K Leung, Christian Makaya, Ting He, and Kevin Chan. Adaptive federated learning in resource constrained edge computing systems. *learning*, 8:9, 2018.

[49] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. *arXiv preprint arXiv:1812.00535*, 2018.

[50] Yinjiao Xing, Eden WM Ma, Kwok L Tsui, and Michael Pecht. Battery management systems in electric and hybrid vehicles. *Energies*, 4(11):1840–1857, 2011.

[51] Xiang-Wu Yan, Yu-Wei Guo, Yang Cui, Yu-Wei Wang, and Hao-Ran Deng. Electric vehicle battery soc estimation based on gnl model adaptive kalman filter. In *Journal of Physics: Conference Series*, volume 1087, page 052027. IOP Publishing, 2018.

[52] Jerry Ye, Jyh-Herng Chow, Jiang Chen, and Zhaohui Zheng. Stochastic gradient boosted distributed decision trees. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 2061–2064. ACM, 2009.

[53] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 2–2. USENIX Association, 2012.

[54] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, 2010.

[55] Kuo Zhang, Salem Alqahtani, and Murat Demirbas. A comparison of distributed machine learning platforms. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–9. IEEE, 2017.