DENSO
Crafting the Core

IARIA

Challenges of Big Data, Artificial Intelligence and Vehicle Data

April 2022
Christian Prehofer

Presenter: Dr. Christian Prehofer

- 15+ years industry experience in telecom, Internet and automotive
  - Currently working at hdmi is
- 10+ years experience in university and applied research labs
  - Lecturer at TU München, supervised multiple PhDs
  - One successful startup on indoor positioning
- About 150 publications, 2 books / monographs
  - 7000 citations according to google scholar
- Recent focus on connected vehicles and Big Data Applications, AI

Big Data for Vehicle Data Analysis

- **Big Data for connected vehicle applications**
  - Vehicles generate enormous amount of data
  - Where to process? **In-vehicle, edge and cloud**
- **Use case driving behavior & energy efficiency**
  - Compute efficiency for every second
  - **Comparison** of Big Data processing options
- **Use case driver status monitoring**
  - Privacy preserving data analysis with federated learning
- **Discussion and Outlook**

Motivation – Big Data and Data Analysis in Automotive

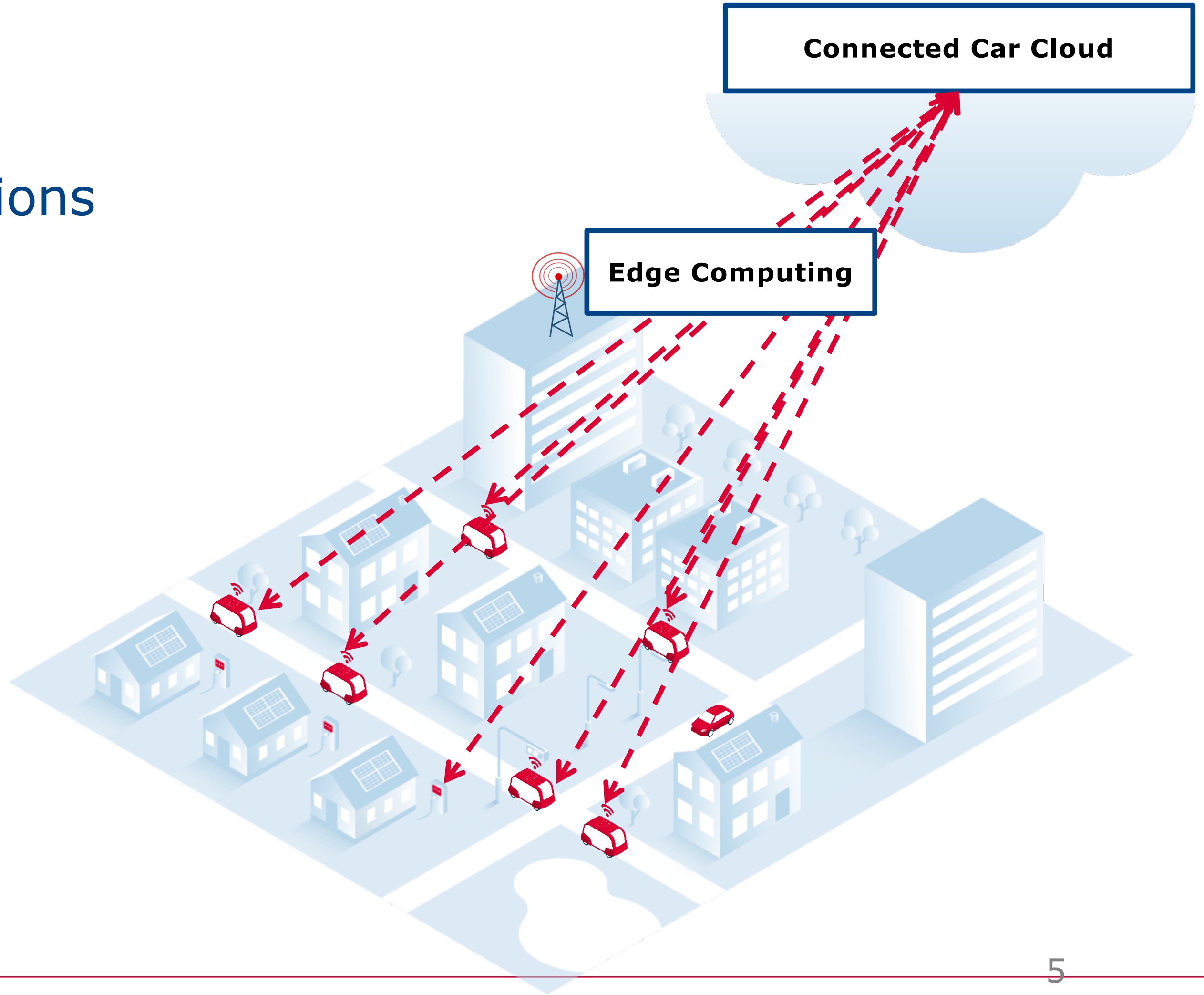
Vehicle Driving Data Applications:
e.g. insurance, eco driving, predictive maintenance, ADAS / AD optimization,…

2TB/day from internal CAN bus

Vehicle Sensor Data: 2 TB/hour

Applications of Connected Vehicles

- **Enhancing in-vehicle functions**
  - Routing and traffic data
  - Energy efficient driving
  - Enhanced autonomous driving functions
- **New services**
  - Insurance based on actual driving
  - Car sharing
  - In-car payment (fuel, …)
- **Management**
  - Predictive maintenance
  - SW / function updates



Source: pixabay

DENSO Crafting the Core

Christian Prehofer, Denso Europe R&D

IoT – Edge – Cloud Overview

Vehicles & IoT

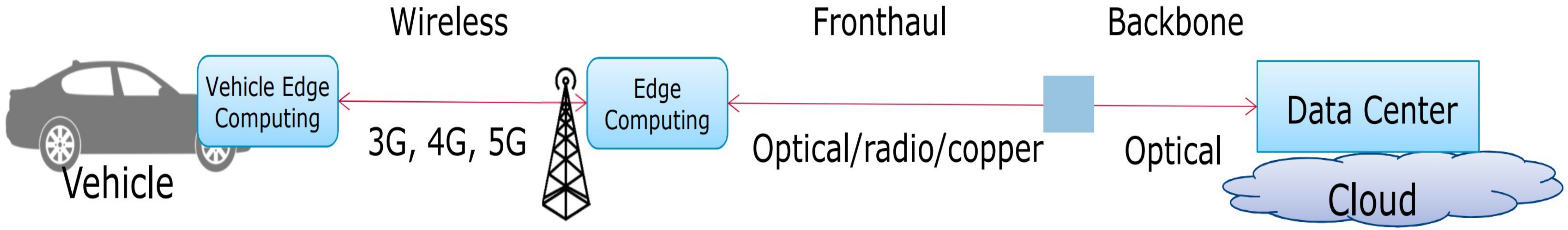
- Lots of sensors & actuators
- Limited computing, cooling needed

Edge Computing:

- Local computing
- Faster response possible (wrt cloud)
- Local data only

Cloud computing

- Fully managed HW / SW, reliability, security
- Economies of scale reg. HW/energy/maintenance/utilization

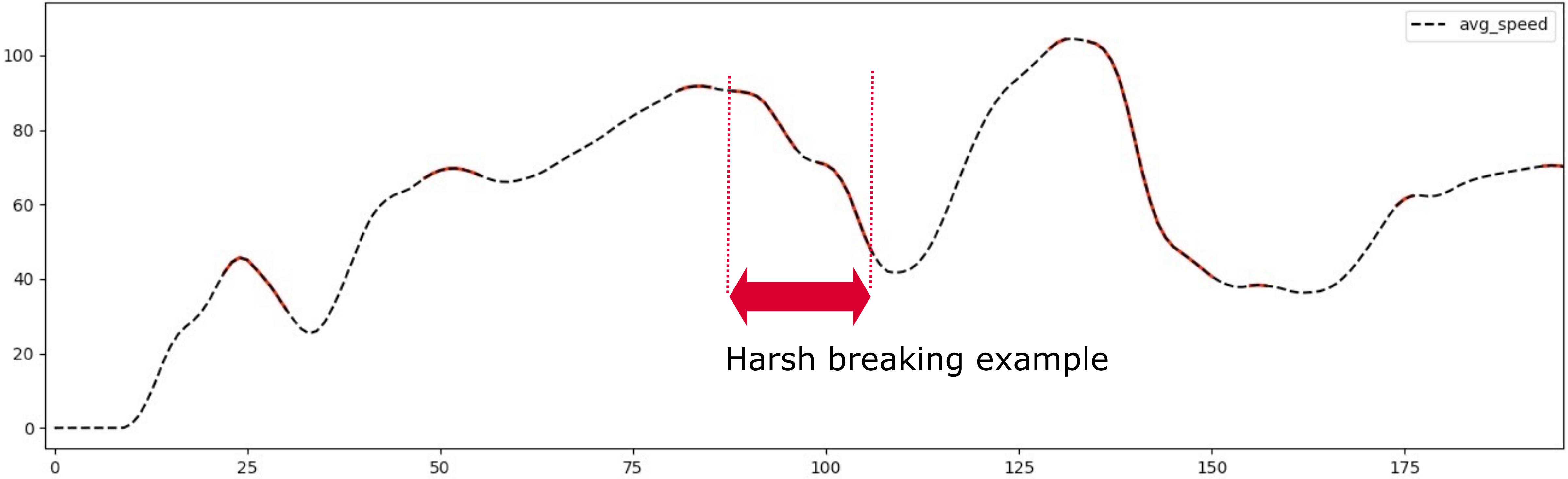


Big Data for Vehicle Data Analysis

- **Big Data for connected vehicle applications**
  - Vehicles generate enormous amount of data
  - Where to process? **In-vehicle, edge and cloud**
- **Use case driving behavior & energy efficiency**
  - Compute efficiency for every second
  - **Comparison** of Big Data processing options
- **Use case driver status monitoring**
  - Privacy preserving data analysis with federated learning
- **Discussion and Outlook**

Example: Harsh breaking

- Find out breaking phases based on speed and acceleration
- Hard brake: deceleration is greater than a certain threshold ….

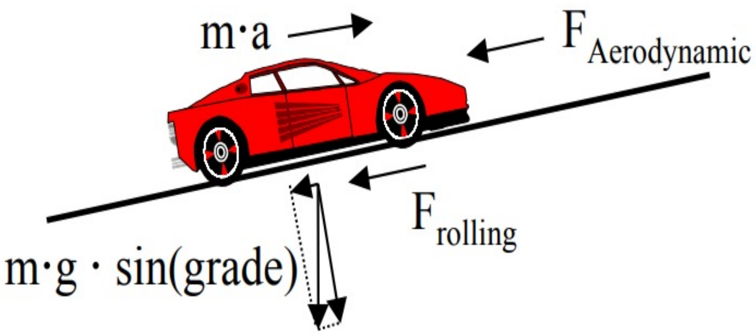


Use Case: Energy Efficiency

https://github.com/gsoh/VED/blob/master/README.md

- Public **data set** (>500 trips, 8000km), incl.
  - Location
  - Speed
  - Energy consumption
  - Air conditioning, heating
  - Vehicle information (weight), …

- Calculate **„needed energy“**
  - **VSP: Vehicle specific power**
  - Need road inclination (from GPS coordinates), acceleration etc



$$VSP\left[\frac{W}{kg}\right] = \frac{Power}{Mass} = \frac{\frac{d}{dt}(E_{kinetic} + E_{potential}) + F_{rolling} \times v - F_{aerodynamic} \times v}{m}$$

Use Case: Comparing Used and Needed Energy

Driving data for e-vehicles

- Speed
- Uphill/downhill
- Vehicle weight

Calculate **needed energy: VSP** (Vehicle Specific Power)

$$VSP\left[\frac{W}{kg}\right] = \frac{Power}{Mass} = \frac{\frac{d}{dt}(E_{kinetic} + E_{potential}) + F_{rolling} \times v - F_{aerodynamic} \times v}{m}$$

Simplified version

$$VSP \approx v \cdot [a \cdot 1.1 + 9.81 \cdot grade + 0.213\,] + 0.000426 \cdot v^3$$

Vehicle **energy consumption**

- KWh from e-vehicle data
- Consider AC and heating
- Temperature, Battery SOC

Compare needed vs actual energy

- Energy efficiency calculation
  - different driving phases
- Energy in different temperatures

Use Case: Energy Efficiency Analysis

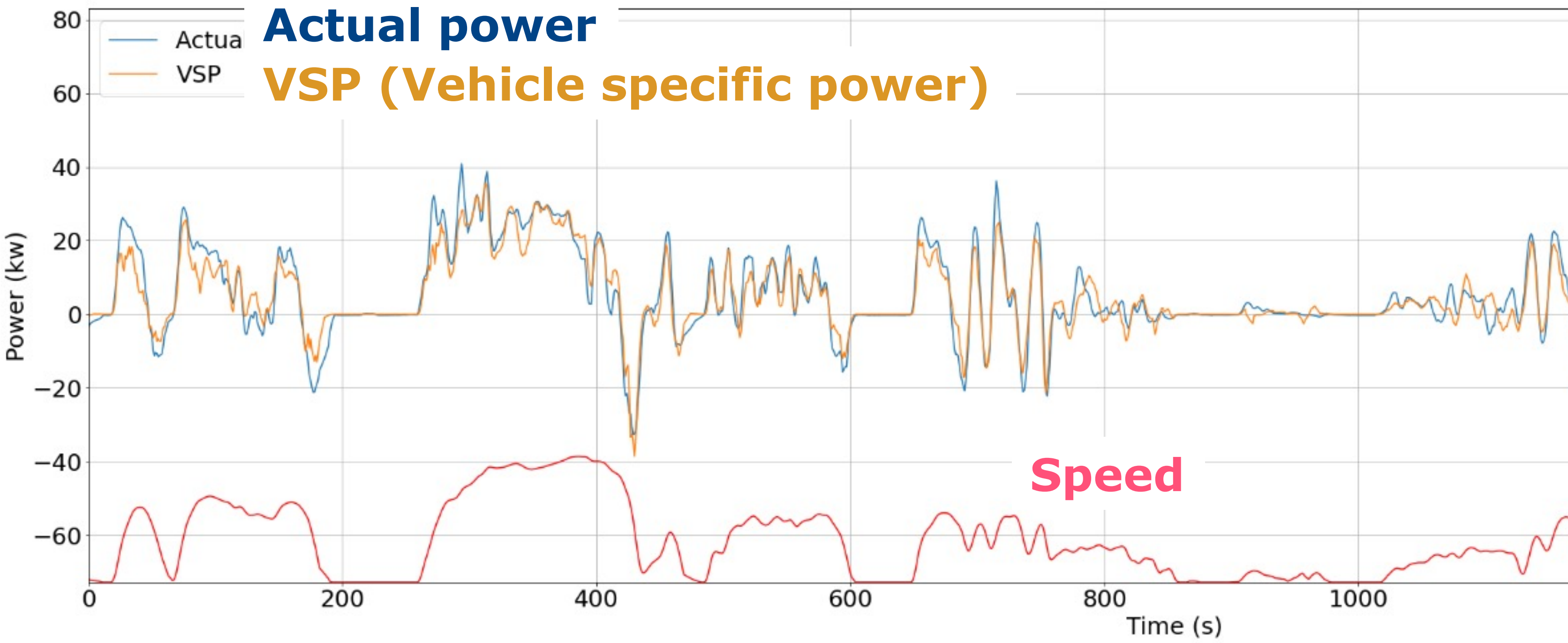
Data Set from E-Vehicles

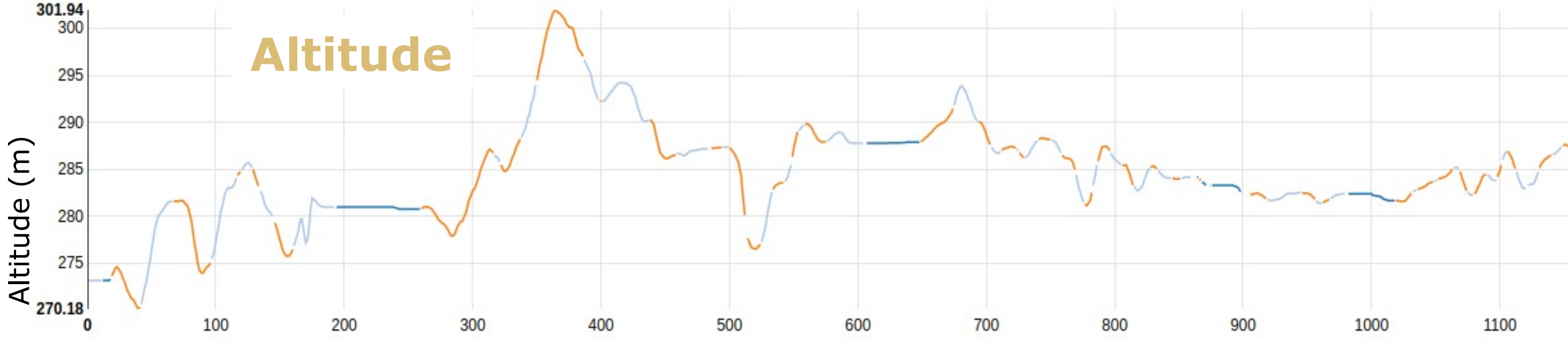
>500 trips, 8000km

Needed vs used Energy

- Calculate physically **energy needed** for movement, **„VSP“**
- Compare VSP to **actual power consumption**, for every second

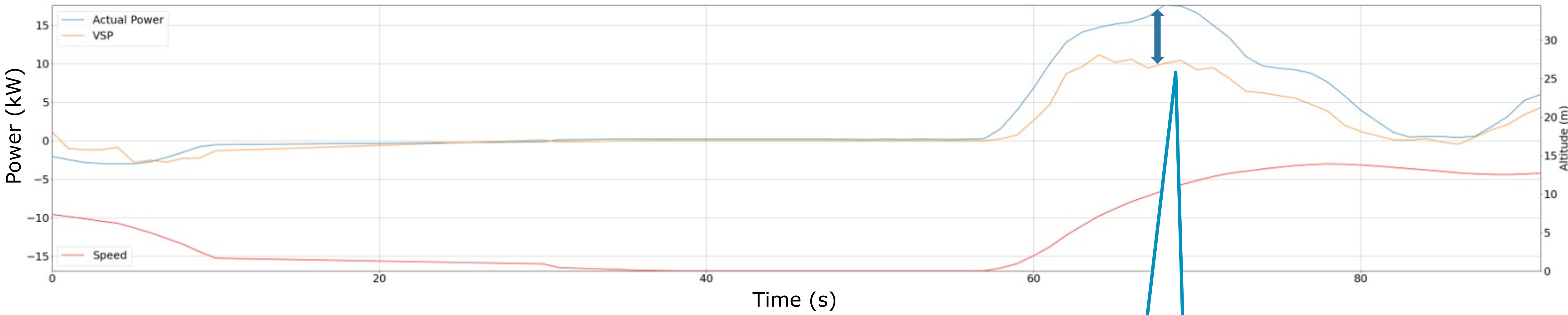
- Evaluation with Apache **Spark, batch processing**





Time (s)

Example in more detail: VSP vs Actual Power

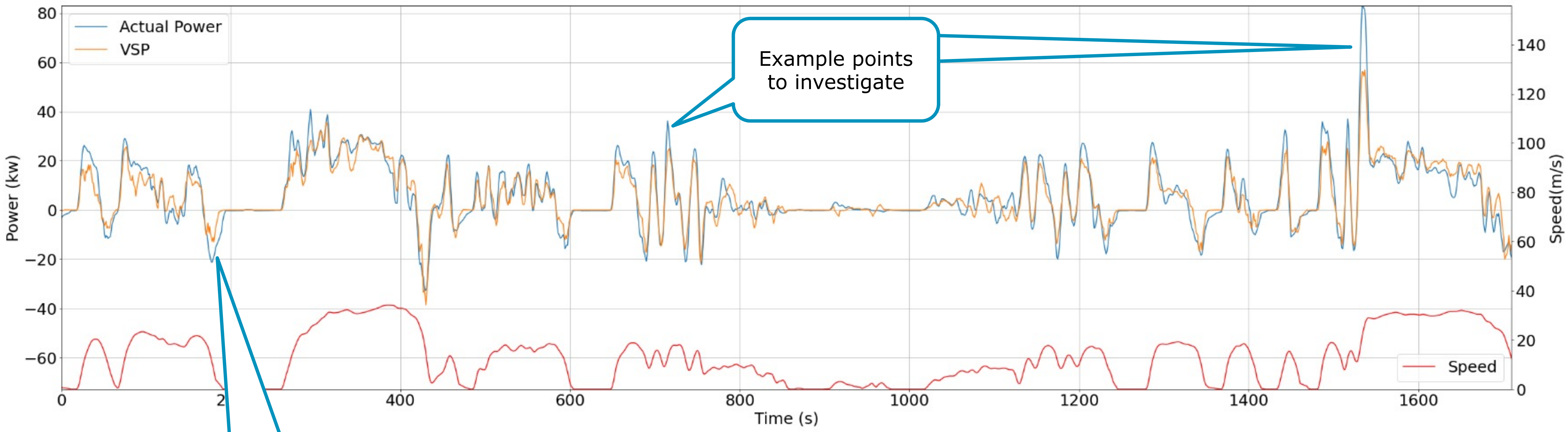


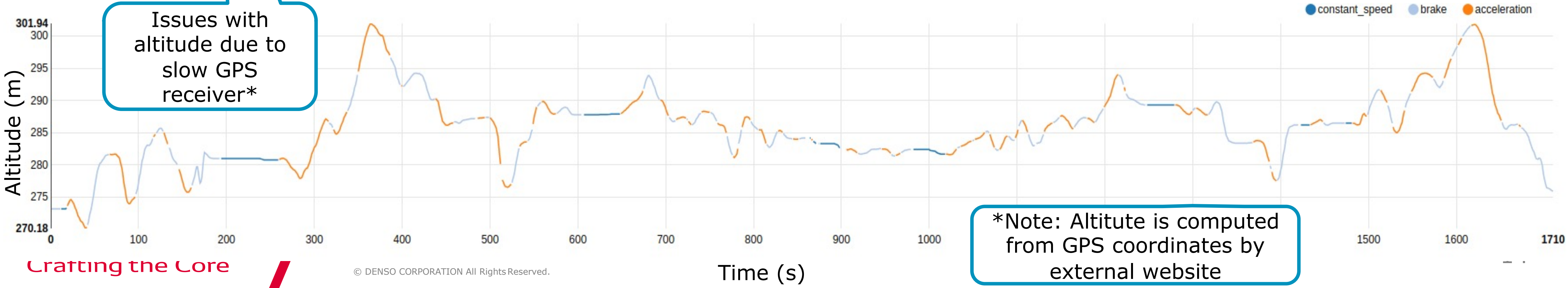
Gaps can indicate inefficiencies or issues

More Details:
Prehofer, C., & Mehmood, S. (2020, December). Big data architectures for vehicle data analysis. In *2020 IEEE International Conference on Big Data (Big Data)* (pp. 3404-3412). IEEE.

E-Vehicle Data with Uphill/Downhill

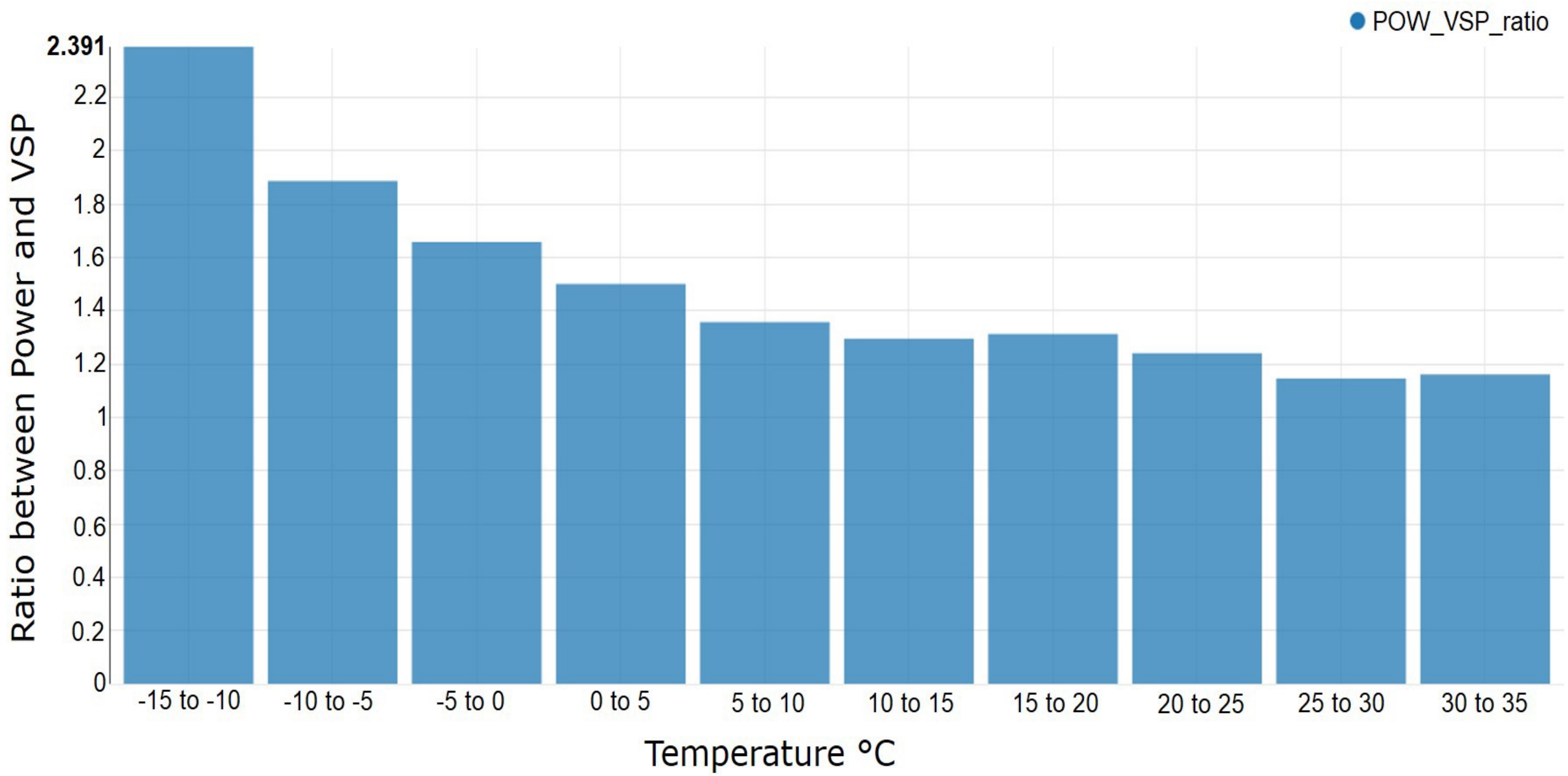
veh_trip_id	total_time	total_power_kW	total_VSP	Correlation(Power,VSP)	max_Temp °C	min_Temp °C	total_airConPow_kW	total_heaterPow_kW	POW:VSP
455_1720	1710.0	10476.059412500792	9865.72836081531	0.9481279556012671	15.0	13.5	0.0	0.0	1.0618637600148808





E-Vehicle Energy Consumption wrt Temperature

- Compute ratio between power and VSP for each a full trip
- Aggregation of 370 trips into temperature bins, total 4731 miles
- Clearly shows efficiency loss for colder temperatures

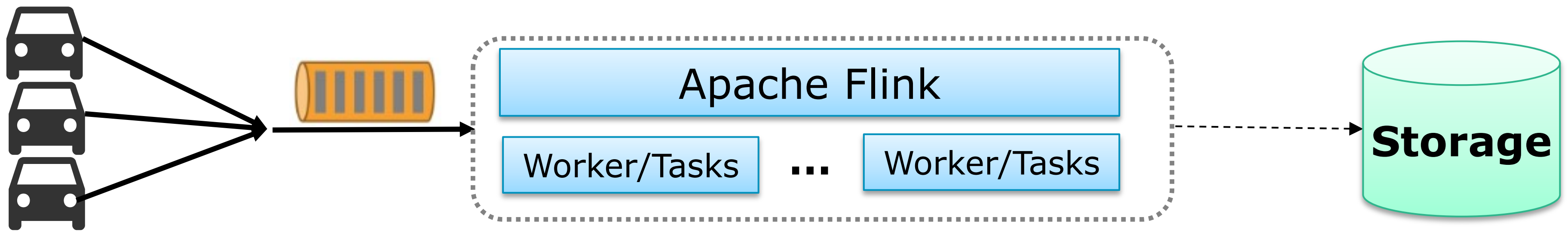


Data Stream processing (for same use case)

Apache Flink Stream processing

- Apache Flink as the true streaming processing engine
- The core of Flink is streams and transformations on dataflows
    - Many APIs, incl DataStream and SQL

- Note: Apache Flink mainly designed for online stream processing, Spark for batch.
    - Spark can do stream processing (with micro-batches), Flink can do batch



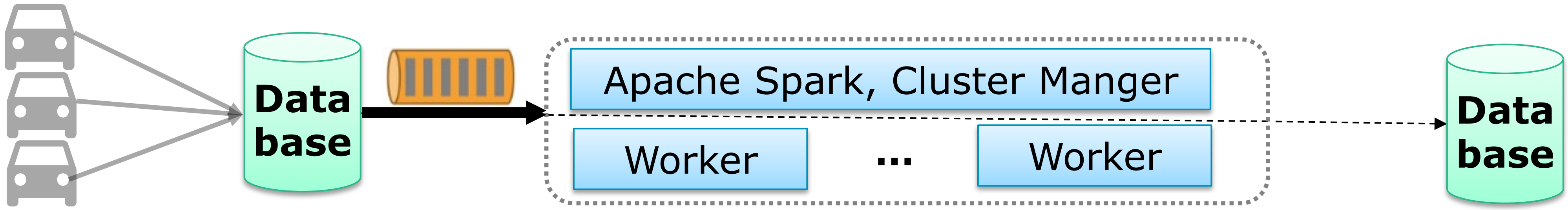
Data Stream processing

Apache Spark Big Data processing

- The core of Spark is generalized “map reduce”
- Functions operating on Dataframes, highly parallel
- High-level APIs, incl SQL and window operations

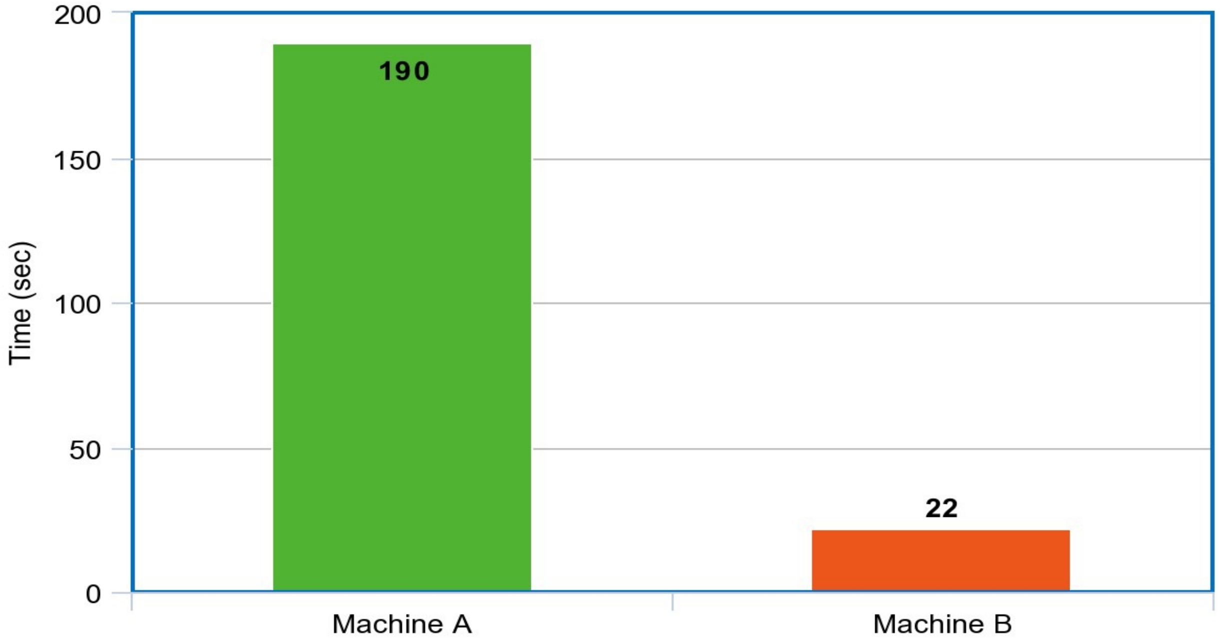
	Spark	Flink
Guarantee	Exactly once	Exactly once
Latency	High	Very Low
Throughput	High	High
Computation model	Micro-batch	Streaming
Overhead of fault tolerance	Low	Low

https://www.ververica.com/blog/high-throughput-low-latency-and-exactly-once-stream-processing-with-apache-flink



Apache Spark Scalability on Similar Use Case

- Implemented with Spark SQL APIs
- Extensive use of window operations
- Stand alone mode
- Graph shows computing time in s



Intel i3, 2 Core, 1.7 Ghz laptop

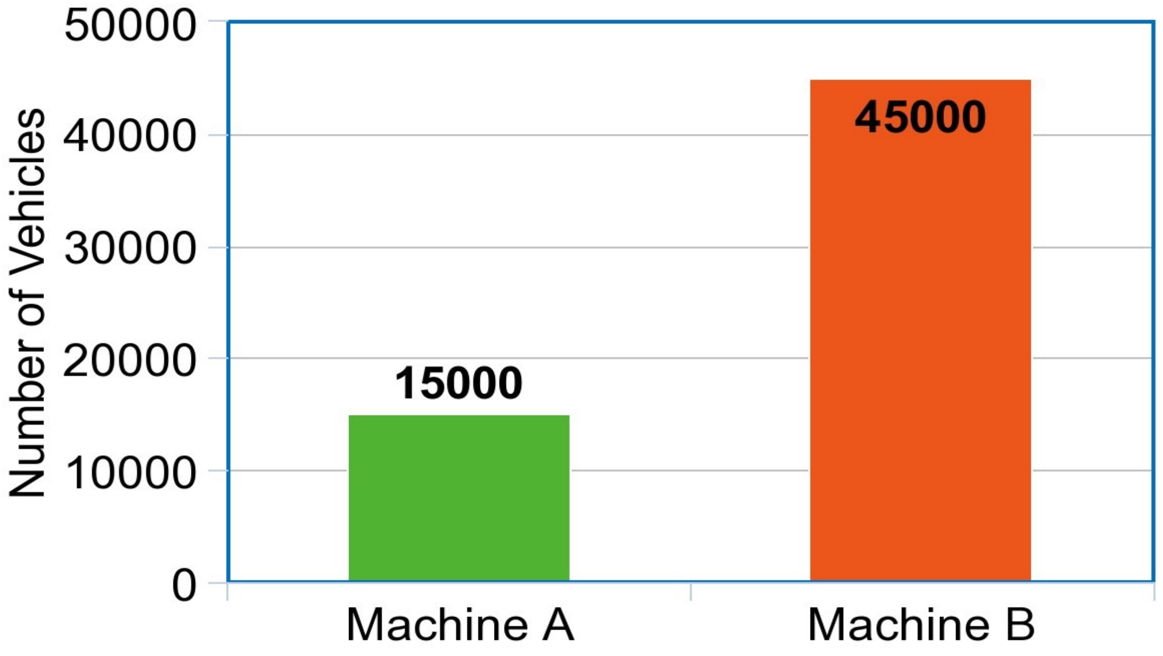
Intel XeonW, 8 Core 3.7GHz, server

More details:

Mohyuddin, S., & Prehofer, C. (2021, February). A Scalable Data Analytics Framework for Connected Vehicles Using Apache Spark. In *2021 International Symposium on Electrical, Electronics and Information Engineering* (pp. 322-329).

Apache Flink Performance on Use Case

- Using Flink DataStream API
  - Expressive window operations
- Stand alone mode



Intel i3, 2 Core,
1.7 Ghz laptop

Intel XeonW,
8 Core 3.7GHz,
server

Spark and Flink Throughput for Vehicle Data Use Case

Throughput for Vehicle Data Use Case	**Spark**	**Flink**
Intel i3, 2 Core 1.7GHz	0,35MB/s	3−4MB/s
Intel XeonW, 8 Core 3.7GHz	3 MB/s	9−12MB/s

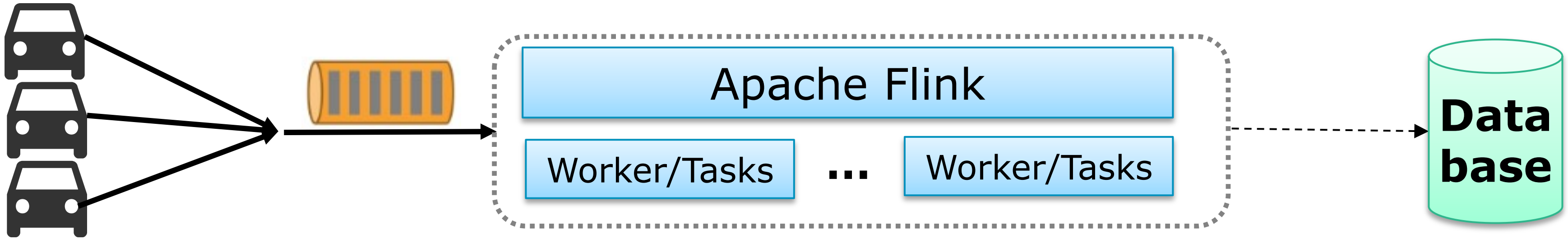
Notes:

- **Performance** can depend significantly on use case and **selected APIs**
  - Flink has powerful window operations in DataStream APIs
  - For Spark, we had to use SQL APIs
- Includes complete execution, incl startup

Christian Prehofer, Denso Europe R&D

Data Stream Processing with Apache Flink

- Apache Flink as the streaming processing engine
- The core of Flink is streams and transformations on dataflows
  - Many APIs, incl DataStream and SQL



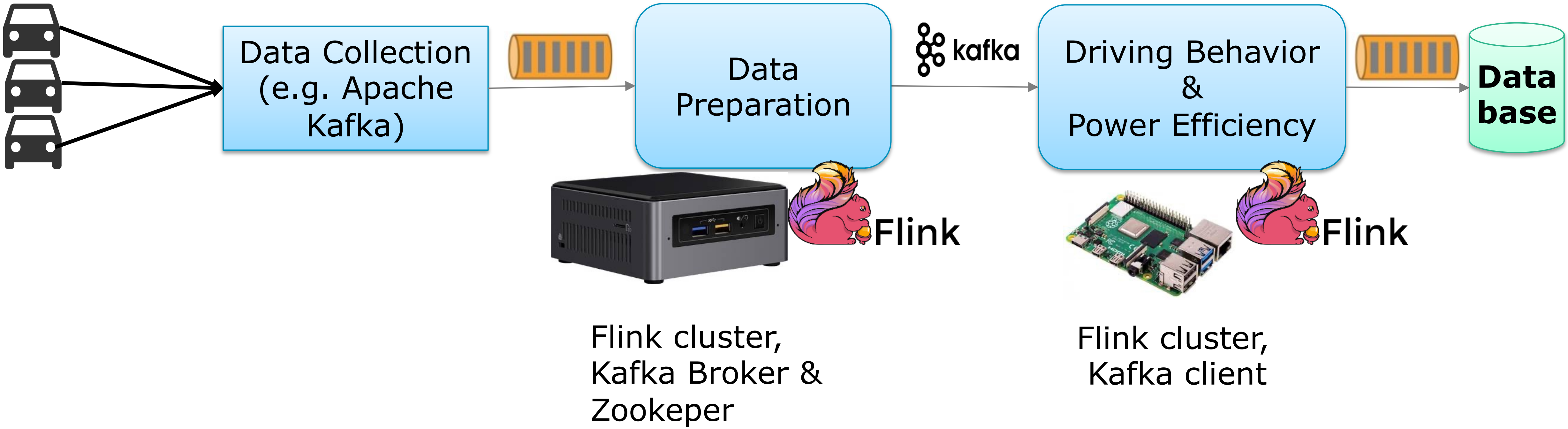
Performance Comparisons: Vehicle data streams

		Scenario 1	Scenario 2
Workstation, Intel Xeon W 3.7GHz, 8 Core, 3000 Euro	Number of Vehicle data streams	**45k**	**60k**
	Avgerage Latency range (ms)	1000 to 1800	1000 to 1800
Raspberry Pi 4b, ARM 7, 1.5 GHz, 4 Cores, 70 Euro	Number of Vehicles	**12k**	**8k**
	Avgerage Latency range (ms)	1000 to 2500	1000 to 3000

DENSO
Crafting the Core

Distributed Processing with Apache Flink - Distributed

Scenario 2, distributed with 2 devices: Using two Flink engines/clusters



Evaluation: Performance on Distributed System

Scenario 2, Distributed	Intel NUC Core i5, 2 Core 2.2GHz	Raspberry Pi 4b, ARM 7, 4 Cores 1.5 GHz
Maximum number of vehicle data streams	**30k**	**30k**
Throughput	6MB/s -9MB/s	6MB/s -9MB/s
Average Latency range (ms)	1000 to 1300	0 to 1200
CPU Utilization %	40% to 70%	60% to 90%

Promising results for 2 Pi 4b‘s!

Data Preparation

kafka

Driving Behavior & Power Efficiency

Big Data for Vehicle Data Analysis

- **Big Data for connected vehicle applications**
  - Vehicles generate enormous amount of data
  - Where to process? **In-vehicle, edge and cloud**

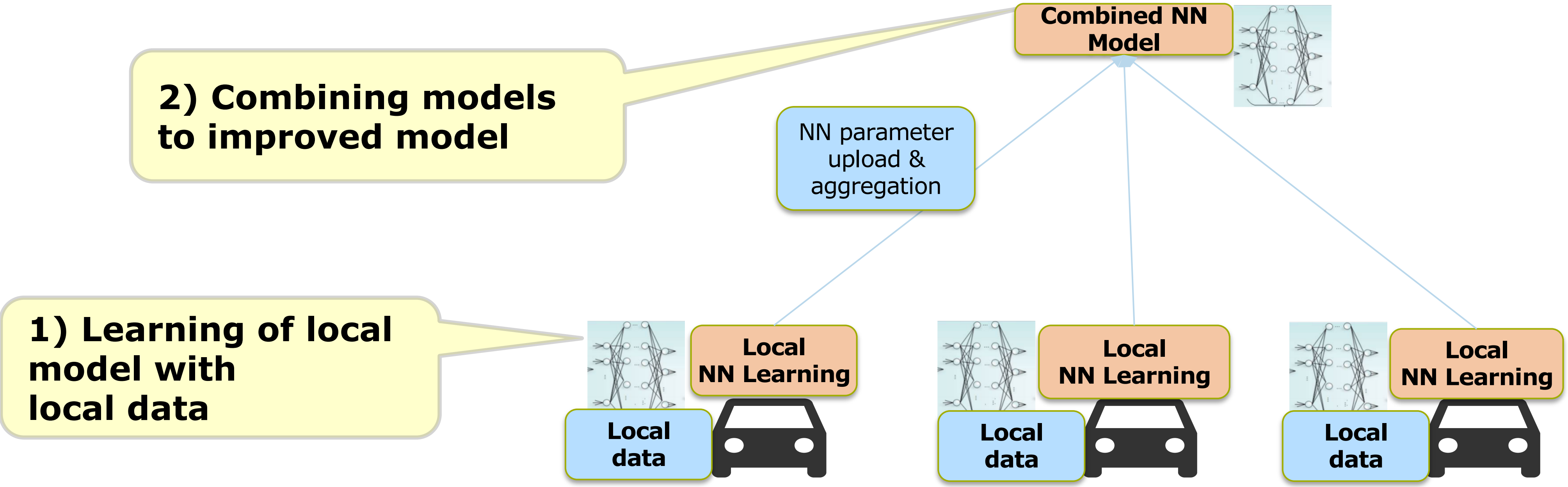
- **Use case driving behavior & energy efficiency**
  - Compute efficiency for every second
  - **Comparison** of Big Data processing options

- **Use case driver status monitoring**
  - Privacy preserving data analysis with federated learning

- **Discussion and Outlook**

Federated Learning Overview

1. Learning with local data in cars to create local model
2. Models are merged from different vehicles/drivers (no image data upload!)
    1. Exchange only NN parameters
3. **Improves privacy + data volume**



Input Data: NTHU Dataset

- 36 people of different genders and ethnicities
- Total 9 and a half hours
  (varying length videos)
- Annotated per frame
  (Eye, Mouth, Head, Drowsiness)
- Train, Val, Test Split (after preprocessing):

*	Training	Validation	Test
Number of Subject	18	4	14
Number of annotations (per-frame)	537,245	145,049	596,590
Number of Videos	288	16	56

[Weng et al., "Driver Drowsiness Detection via a Hierarchical Temporal Deep Belief Network."]

Federated Learning for Driver Drowsiness detection

- **Use case of driver drowsiness detection**
- Detection Method:
  - <u>Using Driver's Behavioral measurements</u> (eyes. mouth, head etc.) non-intrusive
    - Datasets used: **NTHU DDD**, others, e.g. DROZY, UTA-RLDD
    - Features
      - PERCLOS: PERcentage of eye CLOSure
      - FOM: Frequency of Mouth Open

Further Details:
Zafar, A., Prehofer, C., & Cheng, C. H. (2021, September). Federated Learning for Driver Status Monitoring. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)* (pp. 1463-1469). IEEE.

FL: Our Setup

➔ Hyperparameter Selection

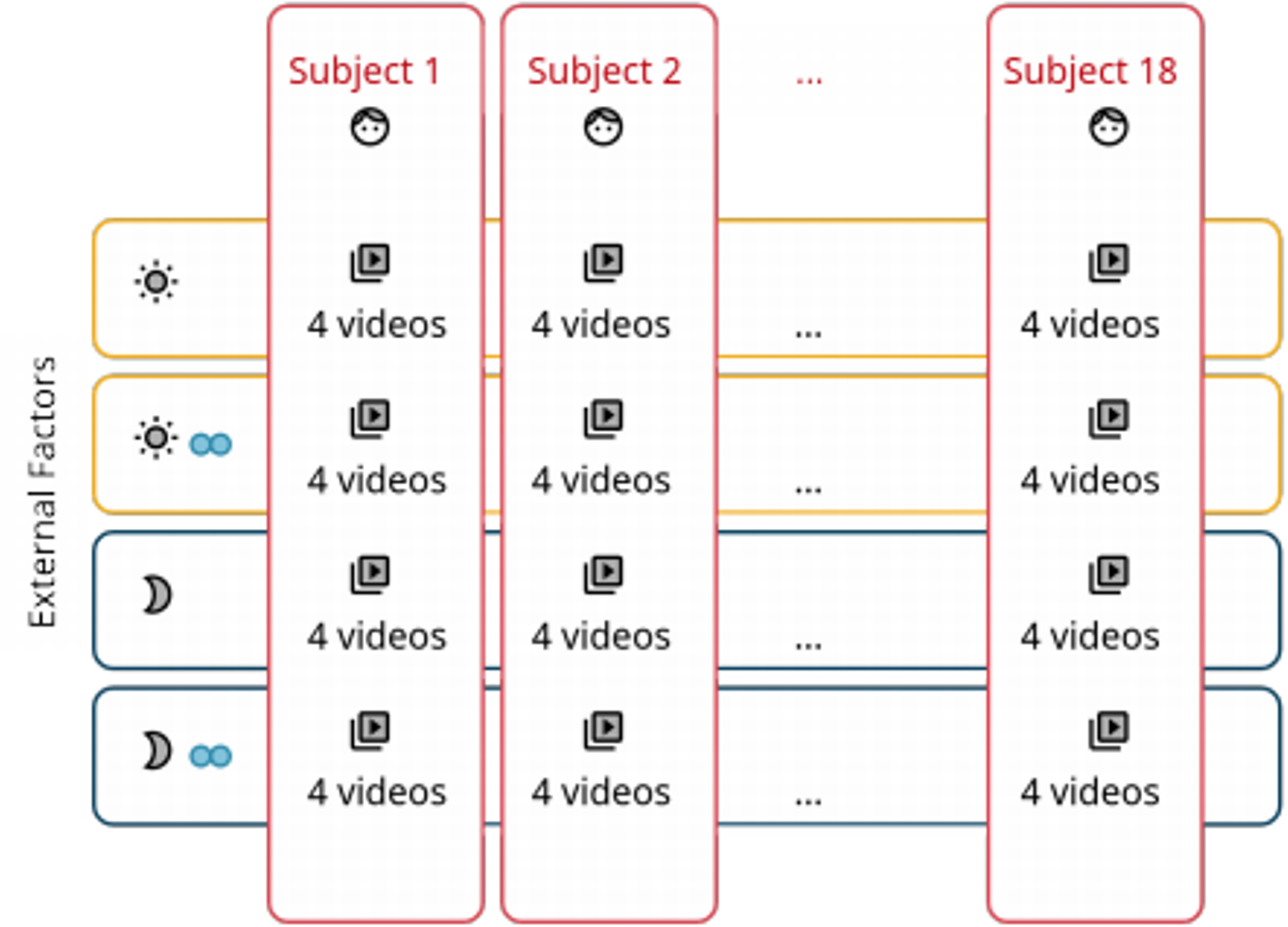
◆K: Total number of clients used the process = 18

◆C: Fraction of clients used at ea iteration = 2

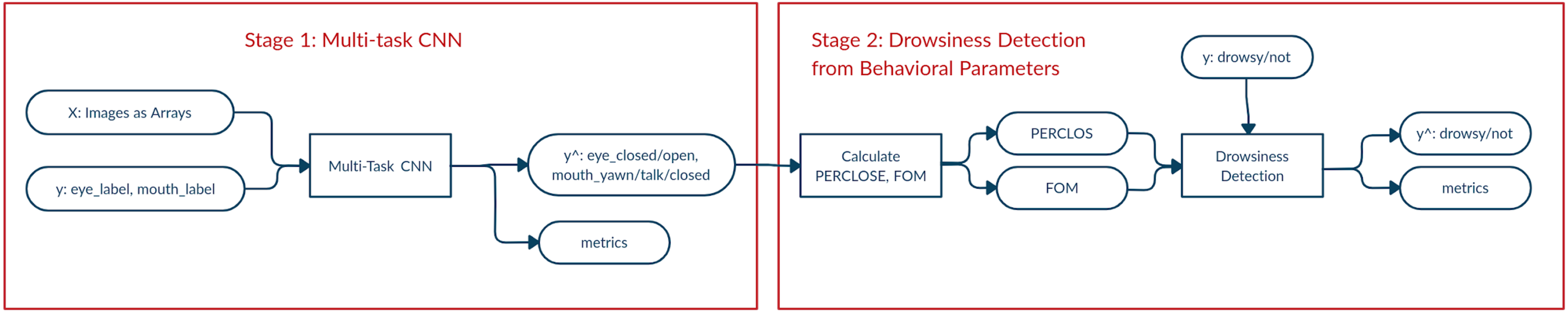
◆Data Sampling

Local dataset at each round: 4 video sequences for varying lengths for a set based on external factors. e.g. night time videos where the driver is wearing glasses

The 4 videos have different facial actions



Baseline Model Architecture



$$PERCLOS = Nclose / Ntotal * 100$$

$$FOM = Nopen / Ntotal * 100$$

Frequency range: All frames in video
Threshold values:
PERCLOS bounded at 20 %
FOM bounded at 16 %

[Savas et al., "Real Time Driver Fatigue Detection System Based on Multi-Task ConNN.",
Zhuang et al. "Driver Fatigue Detection Method Based on Eye States With Pupil and Iris Segmentation"]

Baseline Model Architecture: Stage 1

Input (320 x 240)	
CONV 1 (6 x 316 x 236, kernel: 5x5)	
POOL 1 (maxpool: 2 x 2)	
CONV 2 (8 x 154 x 114, kernel: 5x5)	
POOL 2 (maxpool: 2 x 2)	
CONV 3 (10 x 74 x 54, kernel: 4x4)	
POOL 3 (maxpool: 2 x 2)	
FLATTEN (9990)	
FC (128)	FC (128)
Output (1, Sigmoid)	Output (3, Softmax)

Cross-Entropy Loss

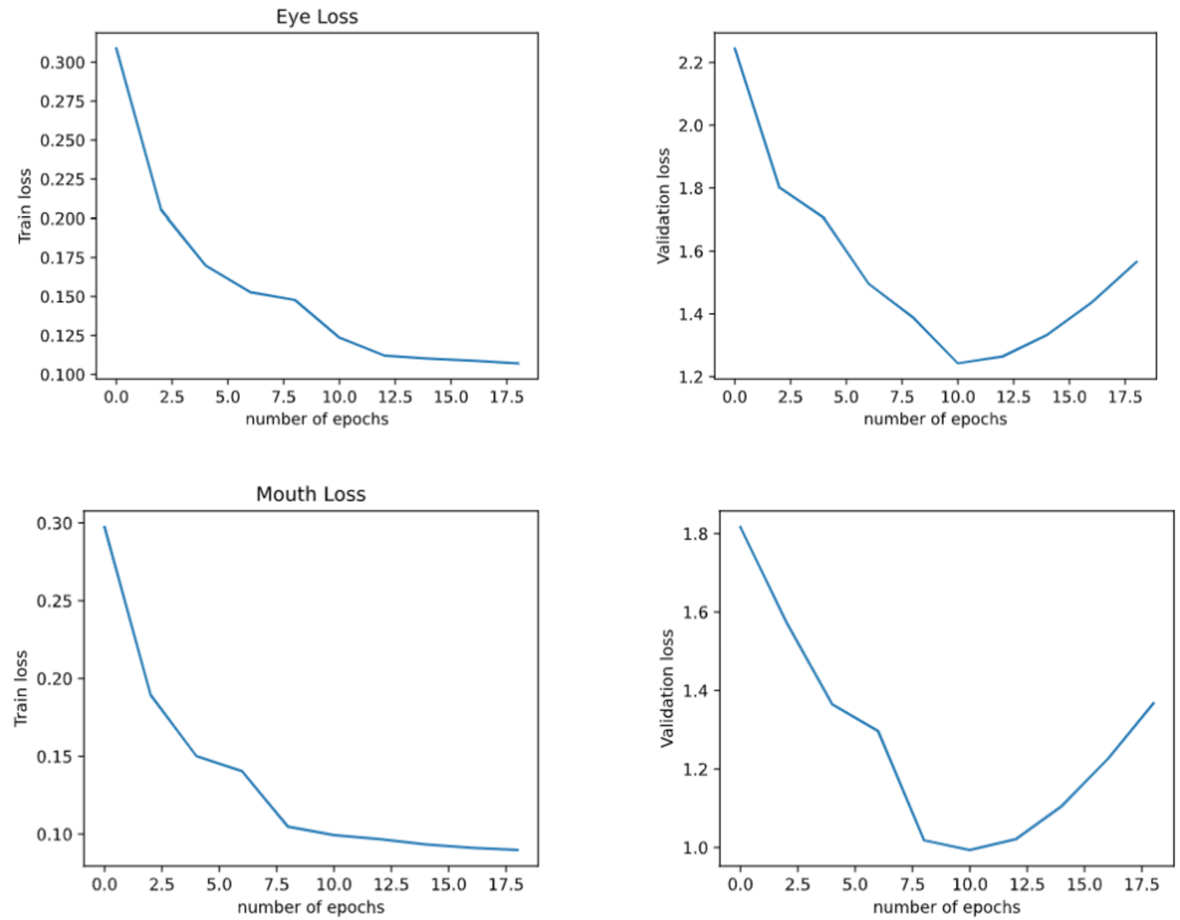
$$-\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log \hat{y}_n + (1-y_n)\log(1-\hat{y}_n)\right]$$

$CEeye$: Binary Cross-Entropy Loss
$CEmouth$: Sparse Categorical Cross-Entropy Loss
$Total\ Loss$:

$$L(y^{\wedge}, y) = CEeye + CEmouth$$

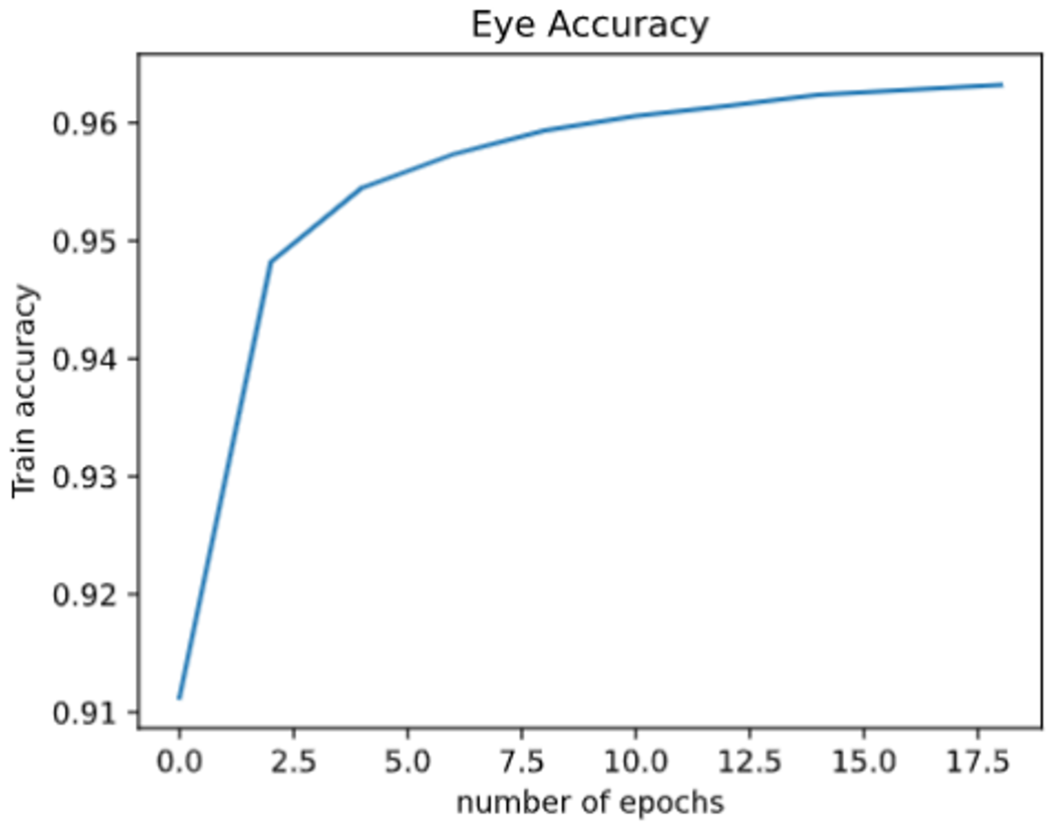
$$Accuracy = (TP + TN)/(TP + TN + FP + FN)$$

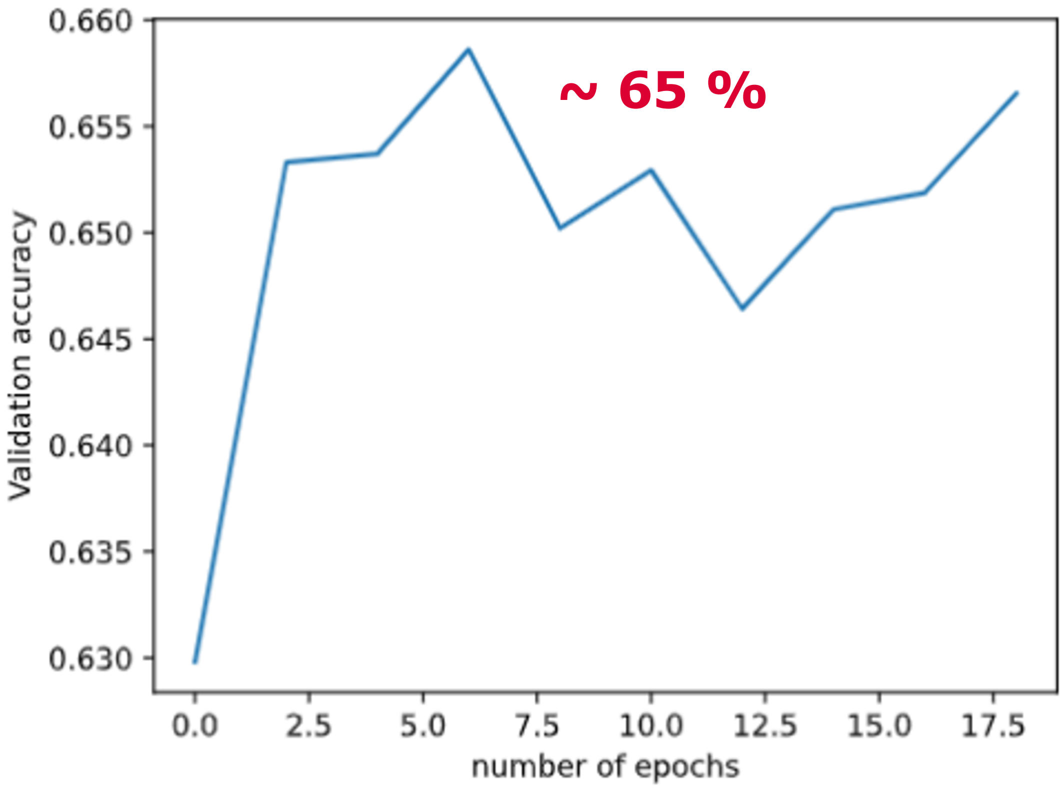
Baseline Model Arch: Loss



Experiment: initial learning rate: 1e −2 (0.01), lr decay: 0.001, momentum: 0.99, batchsize: 64, epochs: 20, batchNorm on conv layers + dropout rate (20%) on fc layers

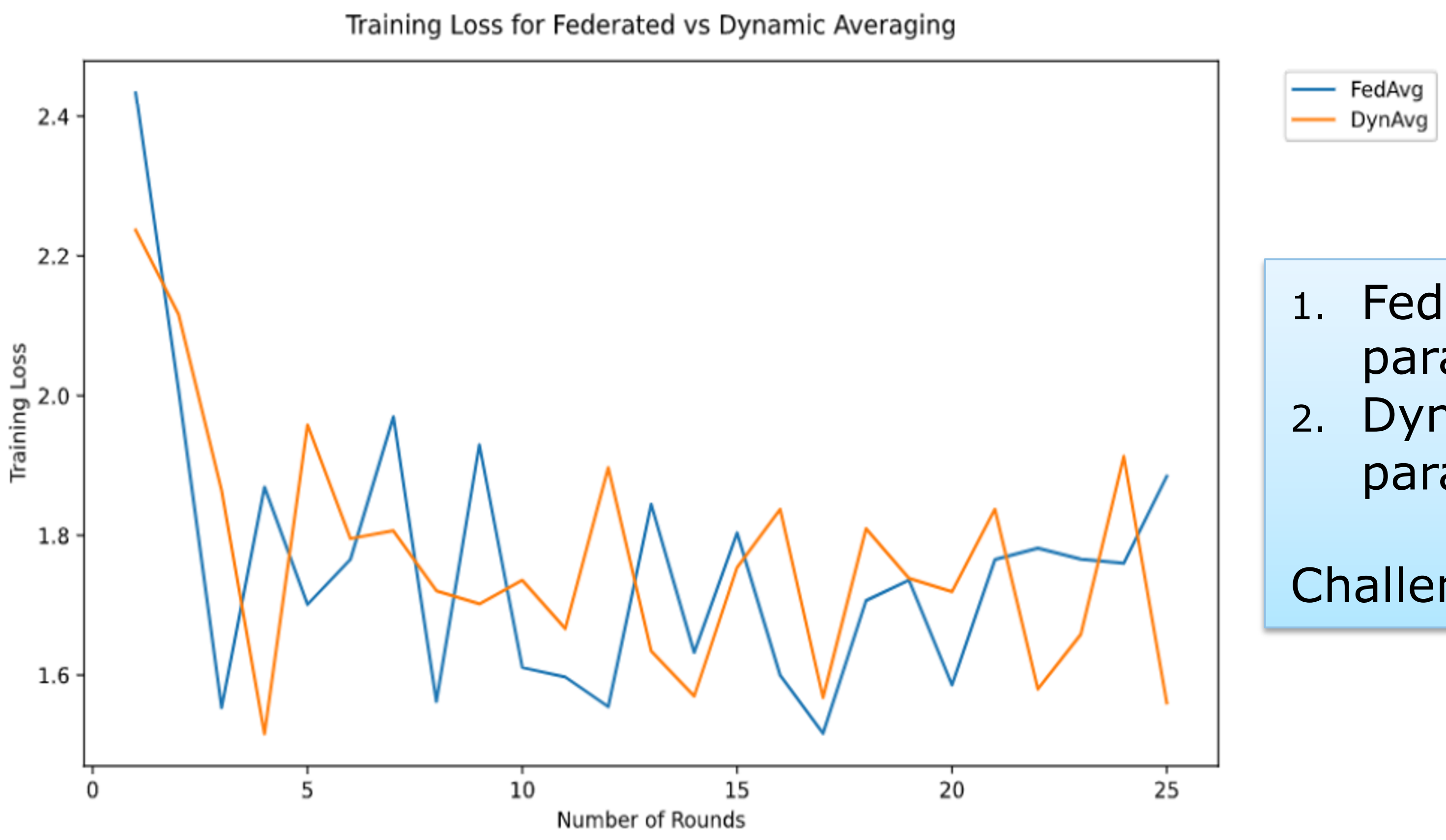
Baseline Model Arch: Final





Experiment: initial learning rate: 1e −2 (0.01), lr decay: 0.001, momentum: 0.99, batchsize: 64, epochs: 20, batchNorm on conv layers + dropout rate (20%) on fc layers

FL: FedAvg and DynAvg



1. Federated Averaging: averaging of all parameters
2. Dynamic Averaging: significant parameter changes updated only

Challenges with highly non-IID data set!

Training loss from our experiment ($\Delta = 0.5$) shows no improvement with non-IID data

[Kamp et al., "Efficient Decentralized Deep Learning by Dynamic Model Averaging"]

FL: how much to aggregate from local updates

Promising Approach: FedProx
Regularization of loss function in local objectives to control divergence

Permit some local deviation from central model

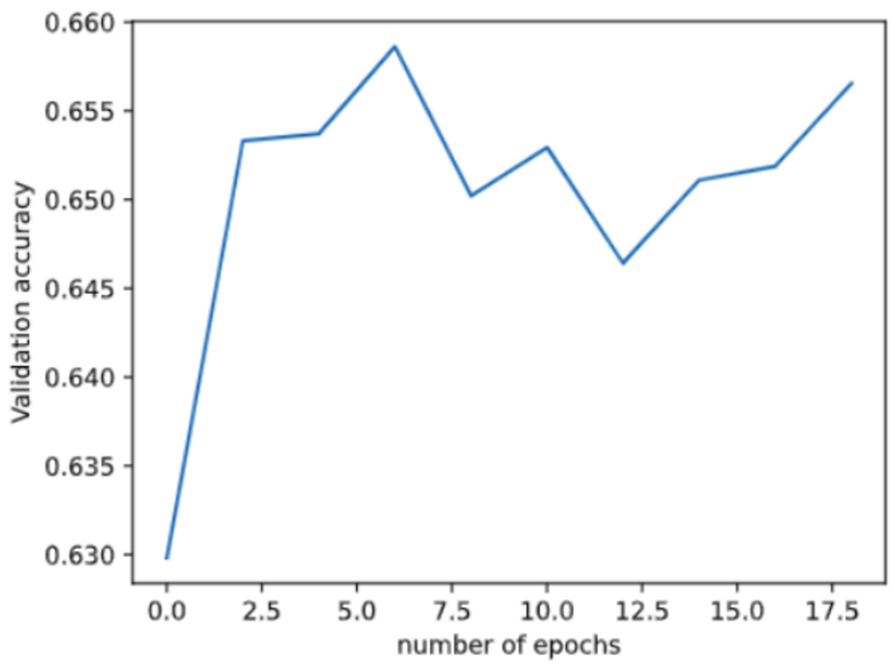
Training loss from our experiment (μ = 0.01) shows improvement

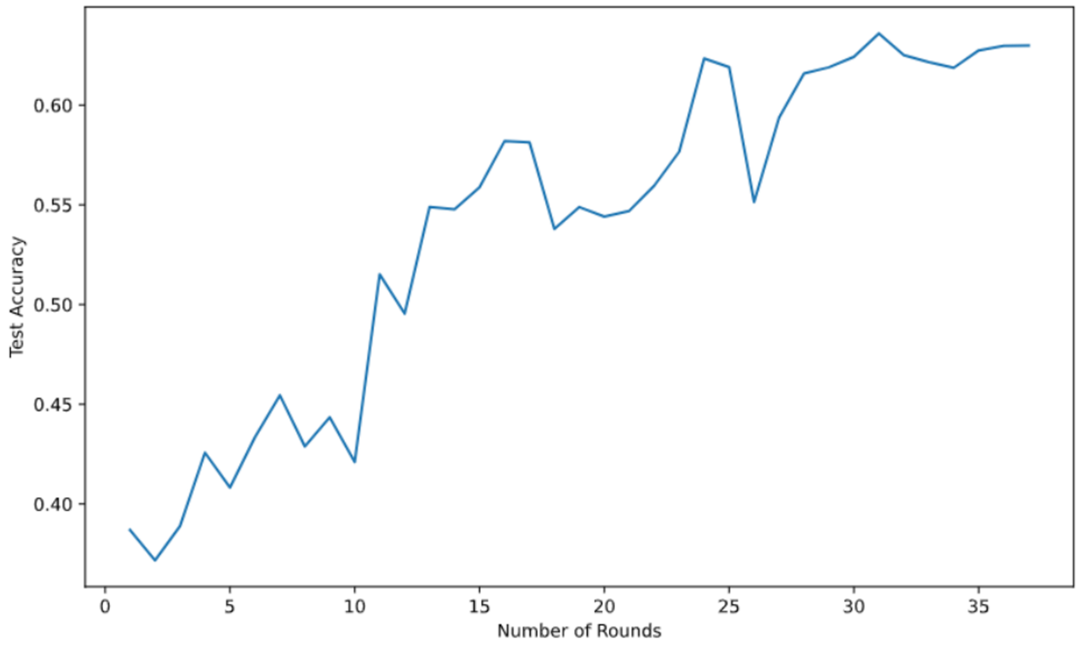
[Li et al., ""Federated Optimization in Heterogeneous Networks"]

Results: Comparing baseline vs FedProx

New results with >80% accuracy in our labs. Needed more and high-quality data

Predictive performance





Test Accuracy for baseline model (65%) and federated model (62%) for Eye Class

Big Data for Vehicle Data Analysis

- **Big Data for connected vehicle applications**
  - Vehicles generate enormous amount of data
  - Where to process? **In-vehicle, edge and cloud**
- **Use case driving behavior & energy efficiency**
  - Compute efficiency for every second
  - **Comparison** of Big Data processing options
- **Use case driver status monitoring**
  - Privacy preserving data analysis with federated learning
- **Discussion and Outlook**

Challenges in Data Science, based on 2017 Kaggle Survey

https://www.kaggle.com/ash316/novice-to-grandmaster

Big Data and Vehicle Data Analysis

Applications for vehicle data with **different requirements**
- Need to understand what data is needed, as well as timing requirements
- Computing in car, in cloud or in the edge

Compared the **architecture options in vehicular** systems,
- Trade-off computing power vs networking vs energy
- Need to consider application development and operation

Performance and scalability of Big Data solutions
- Apache Flink **scales down** to small machines (4 cores)
- **Distributed Big Data** processing can be highly efficient

Privacy-aware distributed AI with federated learning
- Promising first result on FedProx, currently ongoing work

Thanks to Shafqat Mehmood, Atiqa Zafar, Shumail Mohyuddin, Chih-Hong Chen, William Lindskog

Vehicle Energy Dataset (VED)

- About 8000 km of driving data with e-vehicles
  - Detailed data with 1s sample time for speed, energy

TABLE II
CONTENTS OF TIME-STAMPED DYNAMIC DATA

Data Name			Populated %						Sampling Time
			ICE Vehicle & HEV		PHEV		EV		
GPS		Latitude / Longitude (deg)	100 %		100 %		100 %		3 sec
		Vehicle Speed (km/h)	100 %		100 %		100 %		1 sec
Engine Info		Engine RPM (rev/min)	99.98 %		100 %		0 %		2 sec
	Fuel Info	Mass Air Flow (g/s)	100 %	80 %	100 %	50 %	0 %	0 %	2 sec
		Fuel Rate (L/h)		0.3 %		50 %		0 %	5 sec
		Absolute Load (%)		83.9 %		0 %		0 %	5 sec
		Short Term Fuel Trim B1 (%)	89.2 %		15.1 %		0 %		5 sec
		Short Term Fuel Trim B2 (%)	33.0 %		0 %		0 %		5 sec
		Long Term Fuel Trim B1 (%)	79.8 %		15.1 %		0 %		30 sec
		Long Term Fuel Trim B2 (%)	23.8 %		0 %		0 %		30 sec
		Outside Air Temperature (°C)	43.6 %		100 %		100 %		60 sec
Auxiliary Power (HVAC)		AirCon Power (KW)	0 %	0 %	100 %	50 %	100 %	100 %	60 sec
		AirCon Power (W)		0 %		50 %		0 %	
		Heater Power (W)	0 %		19.4 %		100 %		
Battery Info		Battery SOC (%)	0 %		100 %		100 %		60 sec
		Battery Voltage (V)	0 %		100 %		100 %		5 sec
		Battery Current (A)	0 %		100 %		100 %		1 sec

Reference: https://arxiv.org/abs/1905.02081