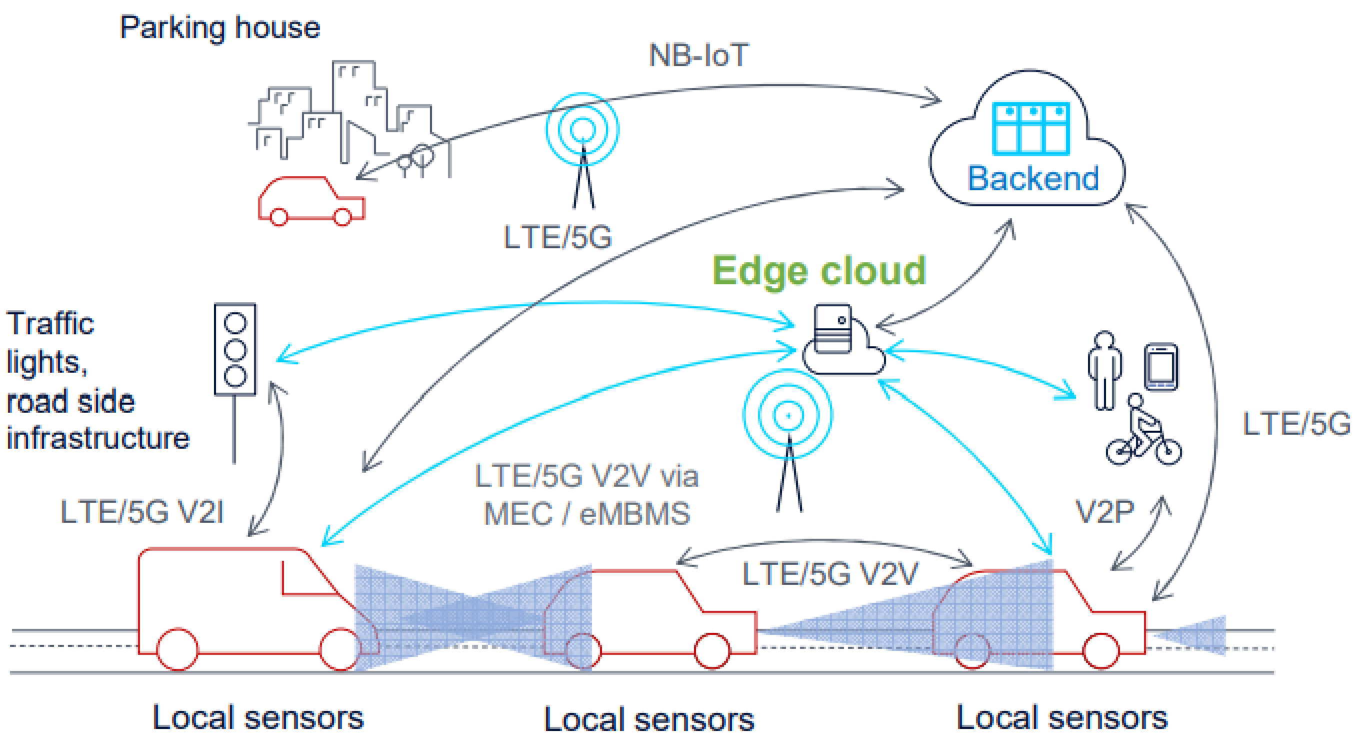
Self-organized systems: Data, learning and decisions

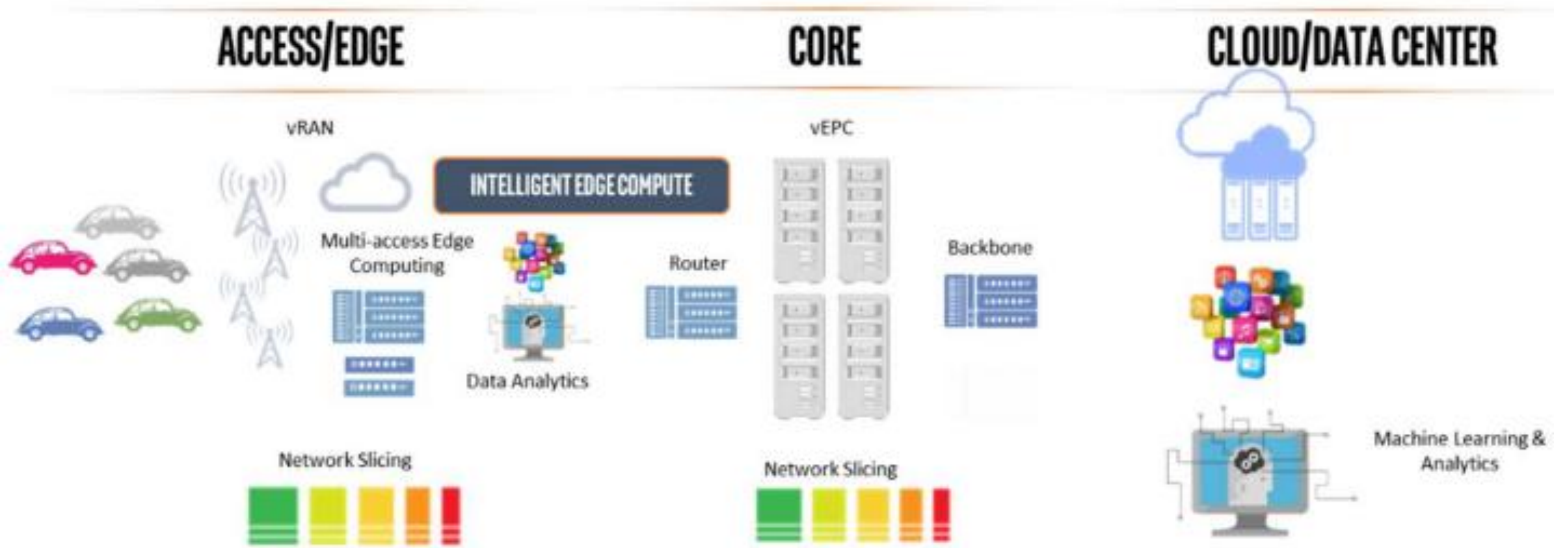
Mestrado em Engenharia de Computadores e Telemática

2024/2025

Use cases and data



Where to process the data?

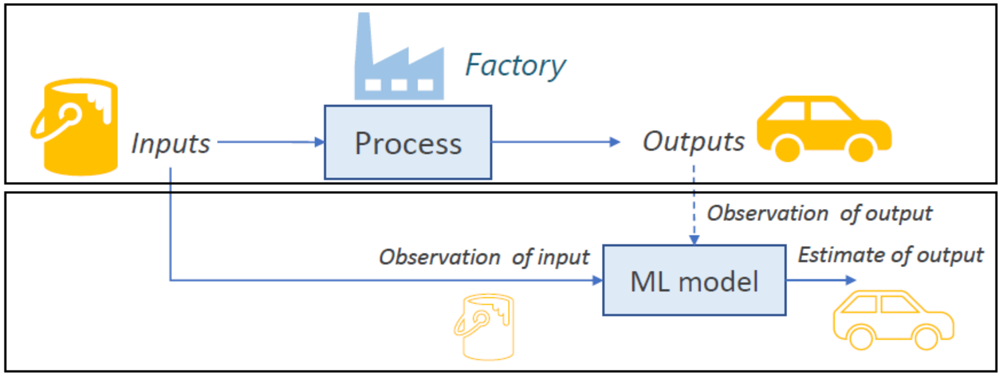


Data Analytics

- Processing data
- Get some decision with the data
- Network decisions with network and services data
  - Give more bandwidth?
  - Assign some special 5G slice to reduce the delay?
- Network decisions with users' data
  - Predict handovers with location and velocity
  - Move the CDN content to the users' most near an access point
  - Ambulance is on the way with network requirements, reserve resources in the network
- User decisions with network data
  - Chose a path with great connectivity for gaming or video
  - Chose a place for remote augmented and virtual reality
- User decisions or robot decisions
  - Obstacle in place?
  - Robot kicks the ball to the right?

Machine Learning

- A pragmatic definition:
  - Collection of algorithms and statistical models (methods) for machines to carry out automated tasks ***based on the observation of inputs and/or outputs of a process***
- The goal of Machine Learning is to produce an estimation or a classification given a set of input values.
- We often distinguish:
  - ML **method**: the **mechanism to train a model** (neural network, support vector machine, etc.)
  - ML **model**: an instance of the method trained to **replicate the behavior** of the target process

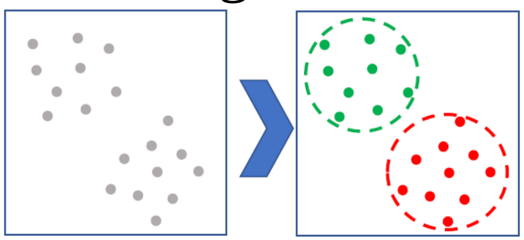


Types of Learning

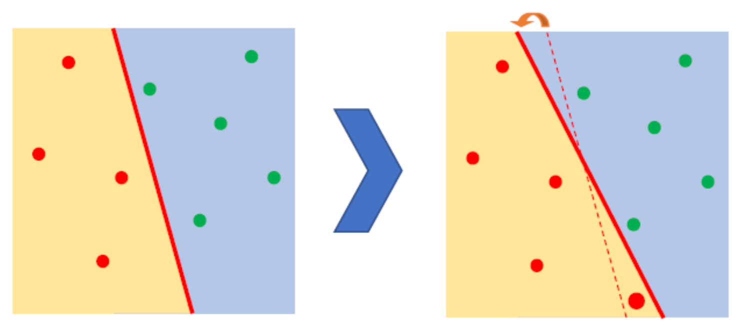
- Supervised: model is trained with a dataset of the target process
  - When training for a classification task, the historical dataset should contain the **Ground Truth** - the actual class of a given sample
- Unsupervised: classification or regression does not depend on prior knowledge

Ground Truth
necessary for training

Historical Dataset		
Feature 1	**Feature 2**	**Class**
A	1	X
B	1	X
C	2	Y

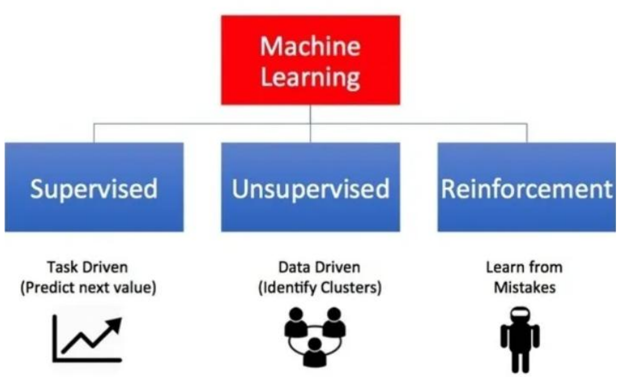


Unsupervised (classes are created by, e.g., finding clusters of similar data points)



Supervised (classification depends on historical inputs)

Types of Machine Learning



Supervised Learning

- Training stage

  A dataset with **input** data and corresponding **output**

  The input data is pre-processed to identify and/or **extract relevant features**

  The **feature data is input to the ML** method, typically one feature set (e.g., mean, median, std. dev.)

  Sometimes a blind set of features is produced, and then only the most relevant are selected (e.g., decision trees)

  For **each input set**, the method produces an **estimate** likely to have an error.

  The method **compares the estimate with the actual process output** (the Ground Truth), and **updates** the model's internal processes to **improve the accuracy** of the estimates.

  The process is **repeated** until performance of the method is **within acceptable bounds**.

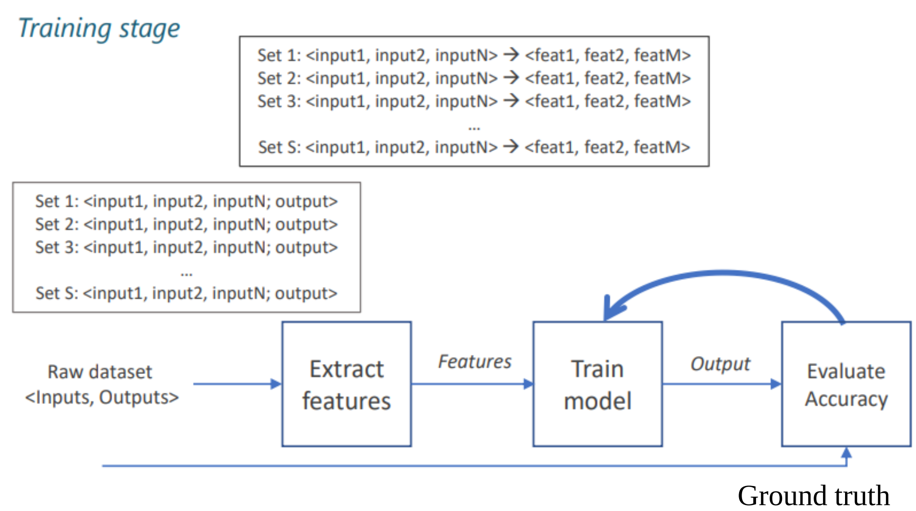
- Inference stage

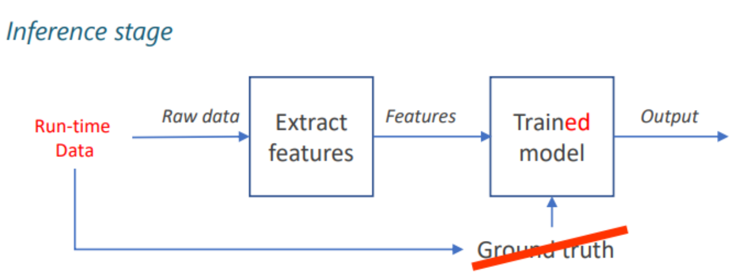
  The trained model is deployed in its target setting.

  Given **inputs**, it can produce **estimates of the process output**.

  However, the method **no** longer has access to the **ground truth**,

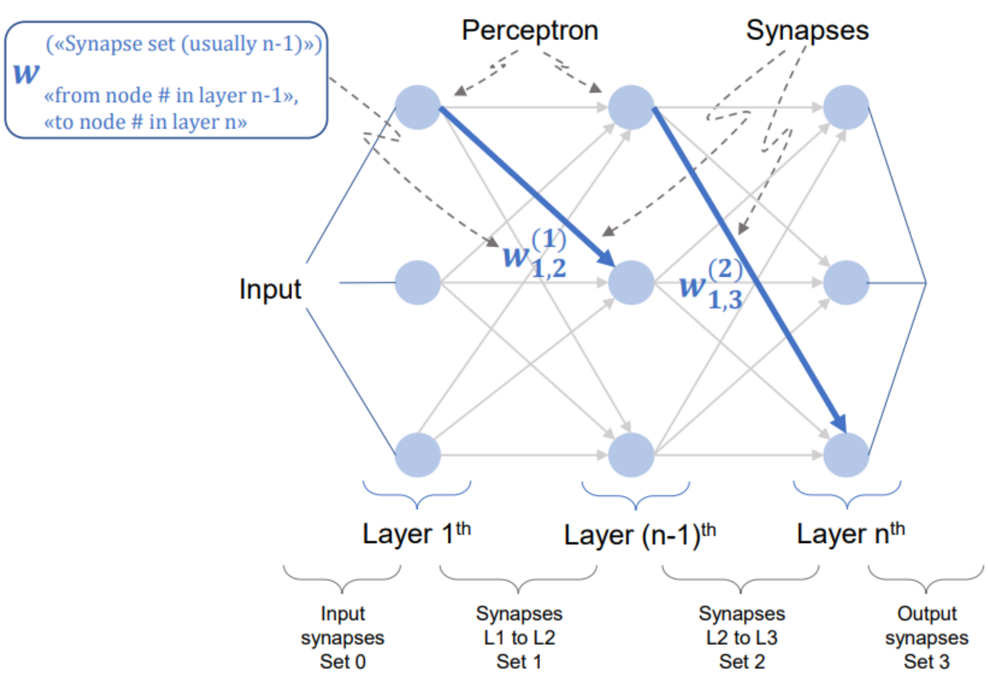
  and it thus enable of further learning.

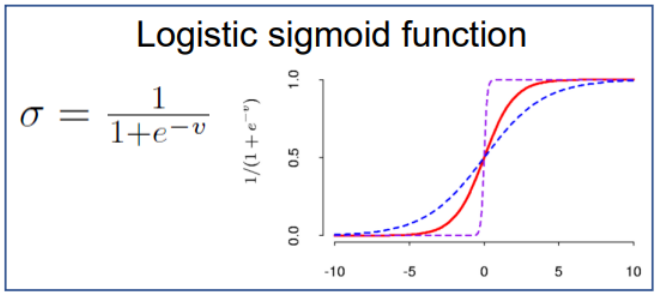




Neural networks

- One of the most successful methods of ML
- Building blocks: **Perceptron** and **Synapse**
- **Perceptron**: typically a **function** that **maps** the entire natural range into a **bounded interval** ([0,1] or [-1,1])
  - Example: **Logistic sigmoid function**, ReLU function, tanh, softmax, etc
- **Synapses**: **connections** from **perceptrons** of layer (n-1)th to perceptrons of layer nth, each applying a **weight** to the transmitted value
- **Training Neural Networks** is mostly about finding the **weights** of those synapses

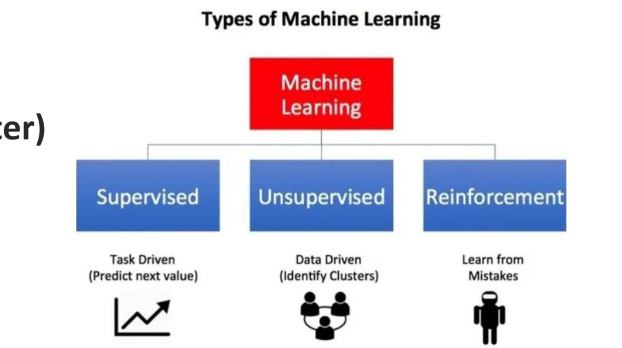




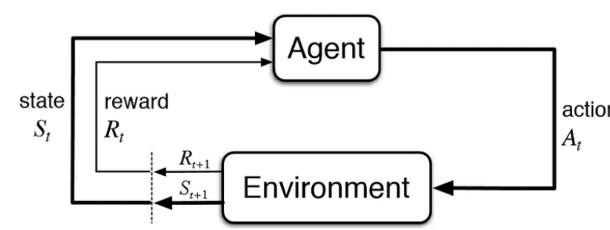
Reinforcement Learning

- Type of machine learning technique that enables an agent to learn in an interactive environment by **trial and error** using **feedback from its own actions and experiences**

1. **Environment —** Physical world in which the agent operates
2. **State —** Current situation of the agent
3. **Reward — Feedback from the environment (good if the next state is better)**
4. **Policy —** Method to map agent's state to actions
5. **Value —** Future reward that an agent would receive by taking an action in a particular state



$$Q(s_t,a_t) \leftarrow (1-\alpha)\cdot \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1},a)}_{\text{estimate of optimal future value}}\right)}^{\text{learned value}}$$



- <u>**Q-learning:**</u> updates Q values which denotes value of performing action *a* in state *s*. The following value update rule is the core of the Q-learning algorithm.

Reward example: best path with resources: path bandwidth / path length

$$reward = \frac{\frac{100.0}{|P|} \times \sum_{l}^{l\in P} R_l}{E_{pl} \times K_{spl}}$$

Learning rate and discount factor:]0 1[

Learning rate: how much to take from previous values

Discount factor: scale down the rewards

Computer games (pacman example), robot automation (RL is used to enable the robot to create an efficient adaptive control system for itself which learns from its own experience and behavior**), …**

Unsupervised learning: K-means

- Centroid: non-data point that indicates center of cluster as identified by K-means

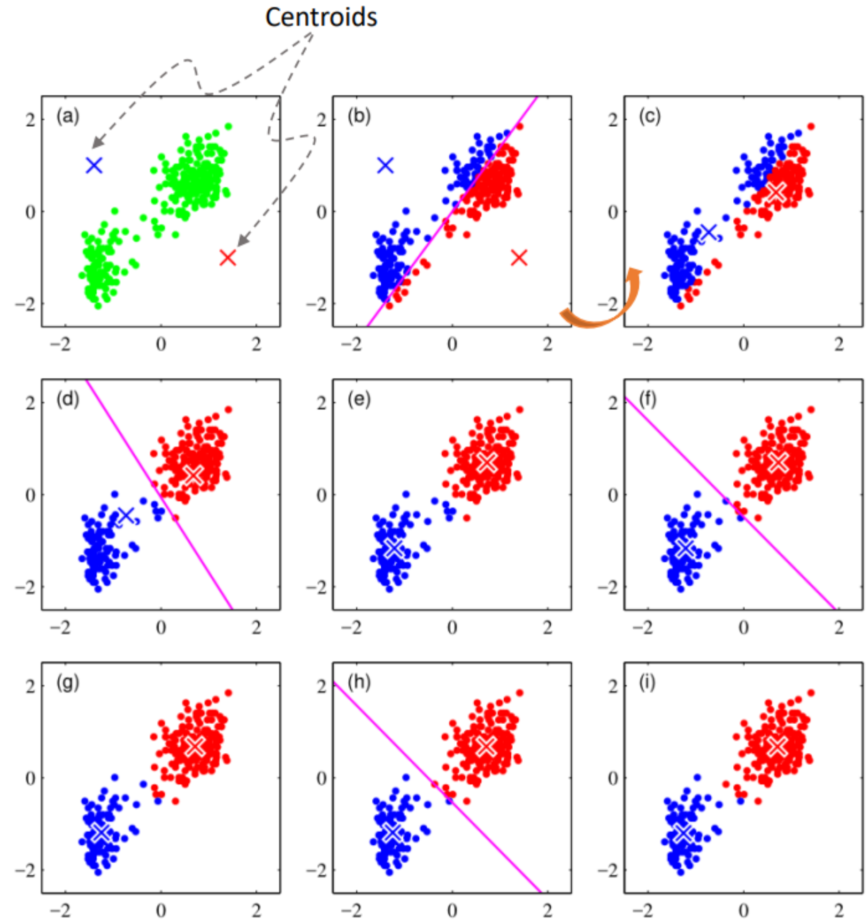
- Operation:

  1. Deploy N centroids randomly (N proportional to number of expected classes)
  2. Assign randomly data points to classes
  3. Repeat iteratively

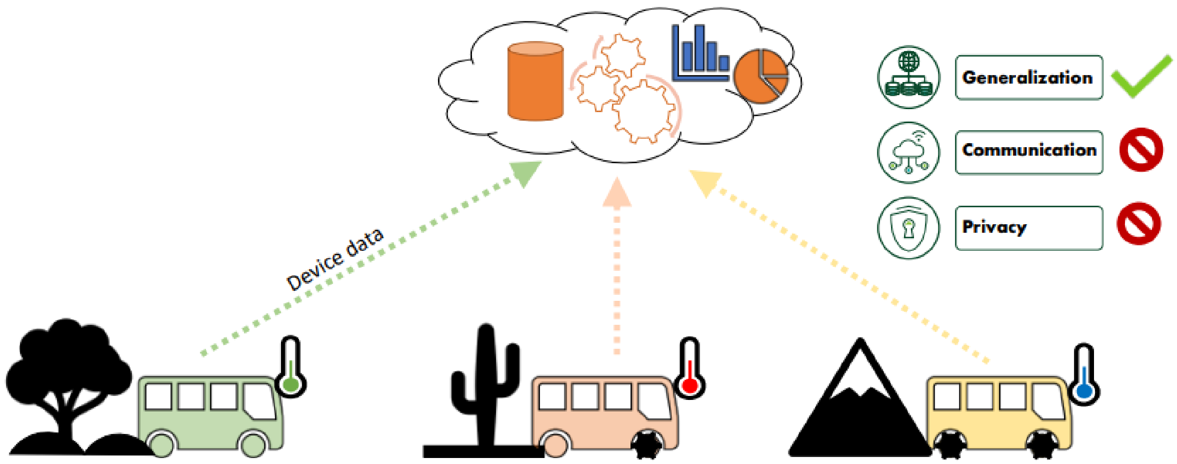
     Minimize metric between points and centroids

     1. Compute center of gravity of each class;
     2. Centroid is repositioned in that center of gravity
     3. Update boundary

  4. Stop when updates become negligible



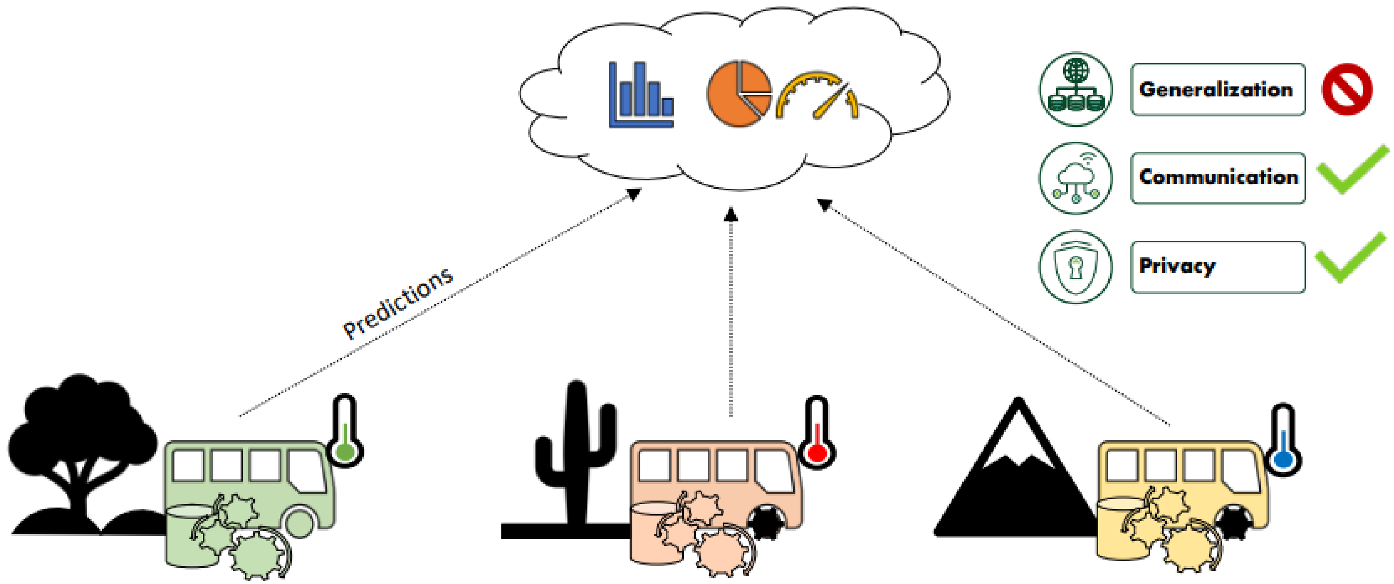
Learning: centralized



Traditional centralized learning – ML runs in the cloud, gathering info from all connected devices and sending back a model.

- The model can generalize based on data from a group of devices and thus instantly work with other compatible devices
- Data can explain all variations in the devices and their environment
- Connectivity - data must be transmitted over a stable connection
- Bandwidth – e.g. a new electrical substation could generate 5 GB/s
- Latency - real-time applications, e.g. automation, requires very low latency
- Privacy - sensitive operational data must remain on site

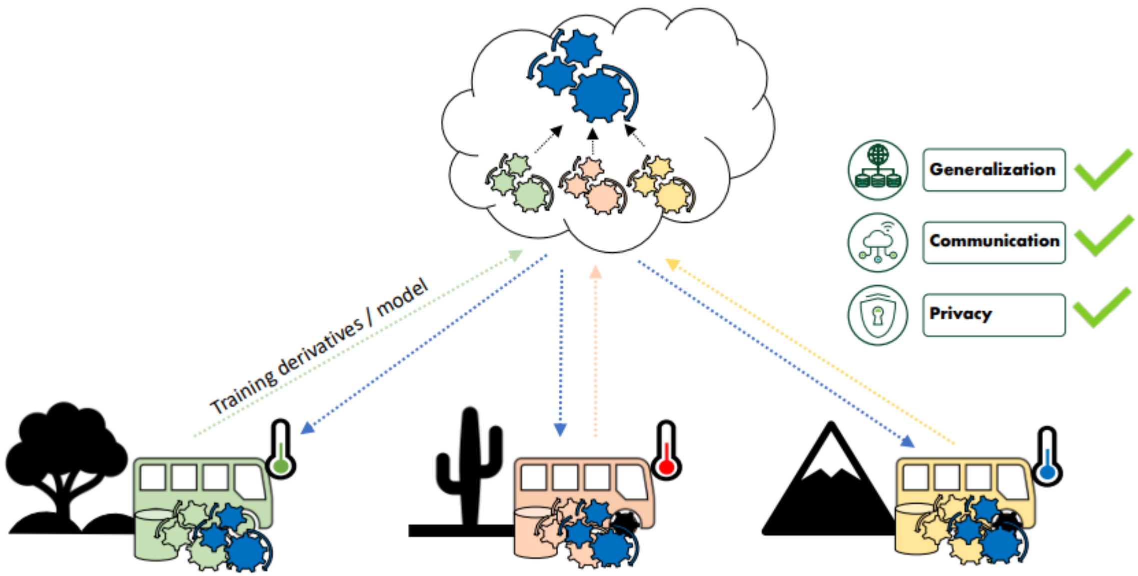
Learning: de-centralized



Edge/decentralized learning: ML continuously onboard each device at the edge of the network.

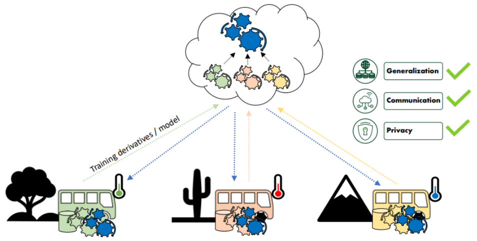
- ML that runs on site, onboard each connected device, by continuously training the ML model on streaming data, the devices learn an individual model for their environment.
- Each model only needs to be able to explain what is normal for itself and not how it varies compared to all other devices.
- Models adapt to changes over time, learning is not constrained by the internet connection and that no confidential information needs to be transferred to the cloud.
- It is not possible to get an overall view and learning

Learning: federated



Federated Learning – learning one from each other while keeping data on the device – creation of Super Models.

- ML technique to train algorithms across decentralized edge devices while holding data samples locally
- Google started as the main player
- Aim to train ML models on billions of mobile phones while respecting the privacy of the users
  - Only send fractions of training results, i.e. training derivatives, to the cloud
  - Never store anything on the device



Federated Learning – learning one from each other while keeping data on the device – creation of Super Models.

Learning: federated

- When collected in the cloud, the partial training results can be assembled to a new supermodel that, in the next step, can be sent back to the devices
    - **Goolgle open source framework TensorFlow Federated**
- Model inspection – evaluation of device behavior through its model
- Model comparison – comparing models in the cloud to find outliers, super models
- Robust learning - learning can continue even if connection to the cloud is lost
- Tailored initialization – new devices can start with a model from a similar device, instead of a general super model

Federated learning: iterative

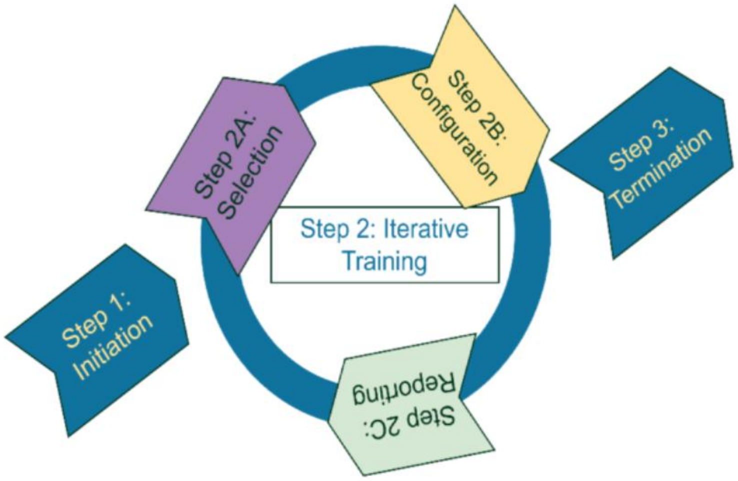
- FL employs an iterative method containing multiple client-server exchanges: federated learning round
  - Diffuse the current/updated global model state to the contributing nodes (participants)
  - Train the local models on those nodes to yield certain potential model updates from the nodes
  - Process and aggregate the updates from local nodes into an aggregated global update so that the central model can be updated accordingly
- FL server is used for this processing

and aggregation of local updates to

global updates

  - Local training is performed by local nodes

with respect to the commands of FL server



Learning: federated

- The FL approach allows for mass processing of data in a distributed way
- It can follow a client-server architecture

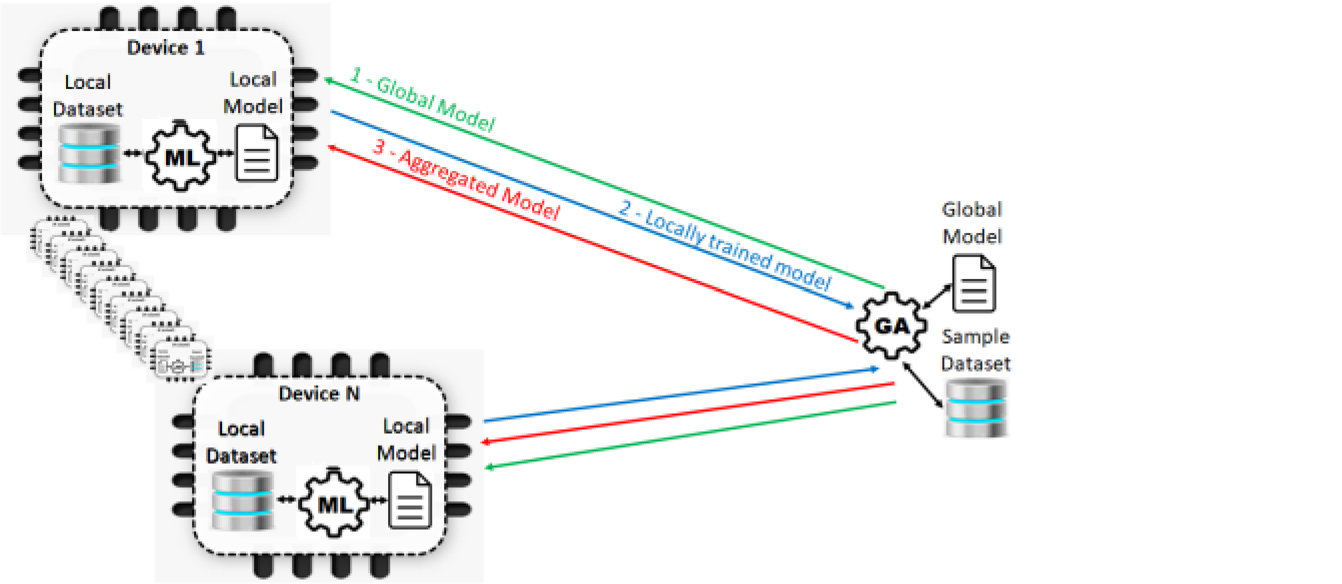
    Server sends the model to be created to the clients (1 – green lines)

    Results of the local computation are sent to the server, which aggregates them into the global model (2 – blue lines)

    Returns the new aggregated model to the clients (3 – red lines)

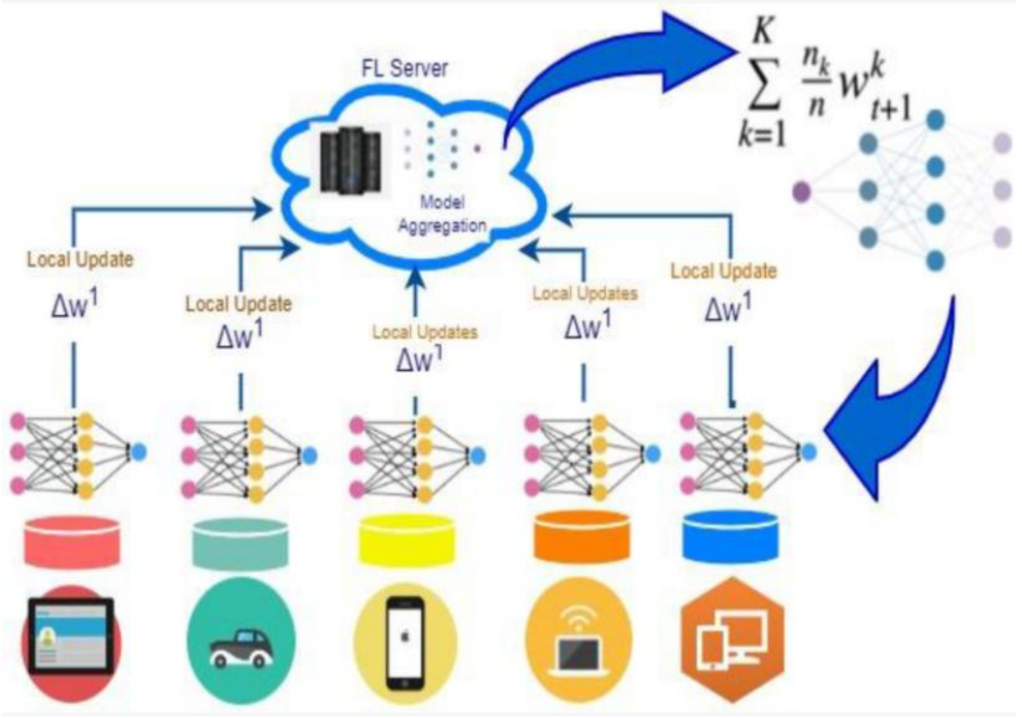
    This iteration, named federated learning round (FLR), occurs until some stopping criterion is reached, such as model convergence or maximum number of iterations reached.

    Edge devices only send information from their local models (parameters, hyperparameters (before training), weights, etc).



Learning: federated

- Federated learning distributes deep learning by eliminating the necessity of pooling the data into a single place
- In FL, the model is trained at different sites in numerous iterations



Learning: model aggregation

- Effective aggregation of distributed models across devices is essential for creating a generalized global model.
- Its efficiency affects precision, convergence time, number of rounds and network overhead.
- Federated stochastic gradient descent (SGD): uses a **single instance of the dataset** to perform the **local training** on the client per round of communication. SGD requires a substantial number of training rounds to produce reliable models. This algorithm is the baseline of federated learning.
- **FedAvg** algorithm starts from the SGD, but each client locally performs a train using the local data at the current model with **multiple steps of SGD before sending the models** back to the server for aggregation

FedAvg reduces the communication overhead required to upload and download the FL model

It requires clients to perform more total computation during training.

Local epoch: one complete pass of the training dataset through the algorithm

Algorithm 1 Algorithm FedAvg. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and n is the learning rate [3].

```
function SERVER-SIDE:
    initialize w0
    for each round t = 1,2,... do
        m ← max(C·k,1)
        S_t ← (random set of m clients)
        for each client k ∈ S_t in parallel do
            w_{t+1} ← ClientUpdate(k, wt)
        end for
        w_{t+1} ← Σ_{k=1}^{K} (n_k/n) w^k_{t+1}
    end for
end function

function CLIENT-SIDE:
    ClientUpdate(k,w): // Run on client k
    B ← (split P_k into batches of size B)
    for each local epoch i from 1 to E do
        for batch b ∈ B do
            w ← w − η∇l(w; b)
        end for
    end for
    return w to server
end function
```

Learning: model aggregation

- Fault Tolerant Federated Average: ability of a computing system to continue working in the event of a failure

  Can tolerate some nodes being offline during secure aggregation

- Q-Federated Average: re-weight the objective in order to achieve fairness in the global model

  Gives higher weights to devices with poor performance

  The network's accuracy distribution becomes more uniform

  Fk(·) to the power of (q+1), q is a parameter that tunes the amount of fairness to impose.

$$\min_{w} \; f_q(w) = \sum_{k=1}^{m} \frac{p_k}{q+1} F_k^{q+1}(w)$$

- Federated Optimization: uses a client optimizer during the multiple training epochs and a server optimizer during model aggregation

  ADAGRAD, ADAM, and Yogi

Algorithm 1 FEDOPT

Input: x_0, CLIENTOPT, SERVEROPT
for $t = 0, \cdots, T-1$ **do**
  Sample a subset $\mathcal{S}$ of clients
  $x_{i,0}^t = x_t$
  for each client $i \in \mathcal{S}$ **in parallel do**
    for $k = 0, \cdots, K-1$ **do**
      Compute an unbiased estimate $g_{i,k}^t$ of $\nabla F_i(x_{i,k}^t)$
      $x_{i,k+1}^t = \text{CLIENTOPT}(x_{i,k}^t, g_{i,k}^t, \eta_l, t)$
    $\Delta_i^t = x_{i,K}^t - x_t$
  $\Delta_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta_i^t$
  $x_{t+1} = \text{SERVEROPT}(x_t, -\Delta_t, \eta, t)$

Algorithm 2 FEDADAGRAD, FEDYOGI, and FEDADAM

Initialization: $x_0, v_{-1} \geq \tau^2$, decay parameters $\beta_1, \beta_2 \in [0, 1)$
for $t = 0, \cdots, T-1$ **do**
  Sample subset $\mathcal{S}$ of clients
  $x_{i,0}^t = x_t$
  for each client $i \in \mathcal{S}$ **in parallel do**
    for $k = 0, \cdots, K-1$ **do**
      Compute an unbiased estimate $g_{i,k}^t$ of $\nabla F_i(x_{i,k}^t)$
      $x_{i,k+1}^t = x_{i,k}^t - \eta_l g_{i,k}^t$
    $\Delta_i^t = x_{i,K}^t - x_t$
  $\Delta_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta_i^t$
  $m_t = \beta_1 m_{t-1} + (1 - \beta_1)\Delta_t$
  $v_t = v_{t-1} + \Delta_t^2$ **(FEDADAGRAD)**
  $v_t = v_{t-1} - (1 - \beta_2)\Delta_t^2 \, \text{sign}(v_{t-1} - \Delta_t^2)$ **(FEDYOGI)**
  $v_t = \beta_2 v_{t-1} + (1 - \beta_2)\Delta_t^2$ **(FEDADAM)**
  $x_{t+1} = x_t + \eta \frac{m_t}{\sqrt{v_t} + \tau}$

TensorFlow Federated

- Open source framework for experimenting with machine learning and other computations on decentralized data.
- Locally simulating decentralized computations into the hands of all TensorFlow users.
  - ML model architecture of our choice
  - Train it locally across data by all users
- Version of the NIST dataset that has been processed by the Leaf project to separate the digits written by each volunteer.

```python
# Load simulation data.
source, _ = tff.simulation.datasets.emnist.load_data()
def client_data(n):
  dataset = source.create_tf_dataset_for_client(source.client_ids[n])
  return mnist.keras_dataset_from_emnist(dataset).repeat(10).batch(20)

# Wrap a Keras model for use with TFF.
def model_fn():
  return tff.learning.from_compiled_keras_model(
      mnist.create_simple_keras_model(), sample_batch)

# Simulate a few rounds of training with the selected client devices.
trainer = tff.learning.build_federated_averaging_process(model_fn)
state = trainer.initialize()
for _ in range(5):
  state, metrics = trainer.next(state, train_data)
  print (metrics.loss)
```

federated_learning_for_image_classification.py hosted with ❤ by GitHub

TensorFlow Federated

- Training an ML model with federated learning is one example of a federated computation
- Evaluating it over decentralized data is another
  - Array of sensors capturing temperature readings
  - Compute the average temperature across these sensors
- Each client computes its local contribution
- Centralized coordinator aggregates all the contributions.

```
@tff.federated_computation(READINGS_TYPE)
def get_average_temperature(sensor_readings):
  return tff.federated_average(sensor_readings)
```

get_average_temperature.py hosted with ❤ by GitHub

Flower: A Friendly Federated Learning Framework (https://flower.dev/)

- Open source framework for experimenting with machine learning and other computations on decentralized data.
- Able to use in containers in a federated framework, FedFramework

Containers:

```
/containers/list
/containers/create/server
/containers/create/client
/containers/start
/containers/stop
/containers/remove
```

List containers:

- `http://10.0.22.37:8000/containers/list`

Server Creation:

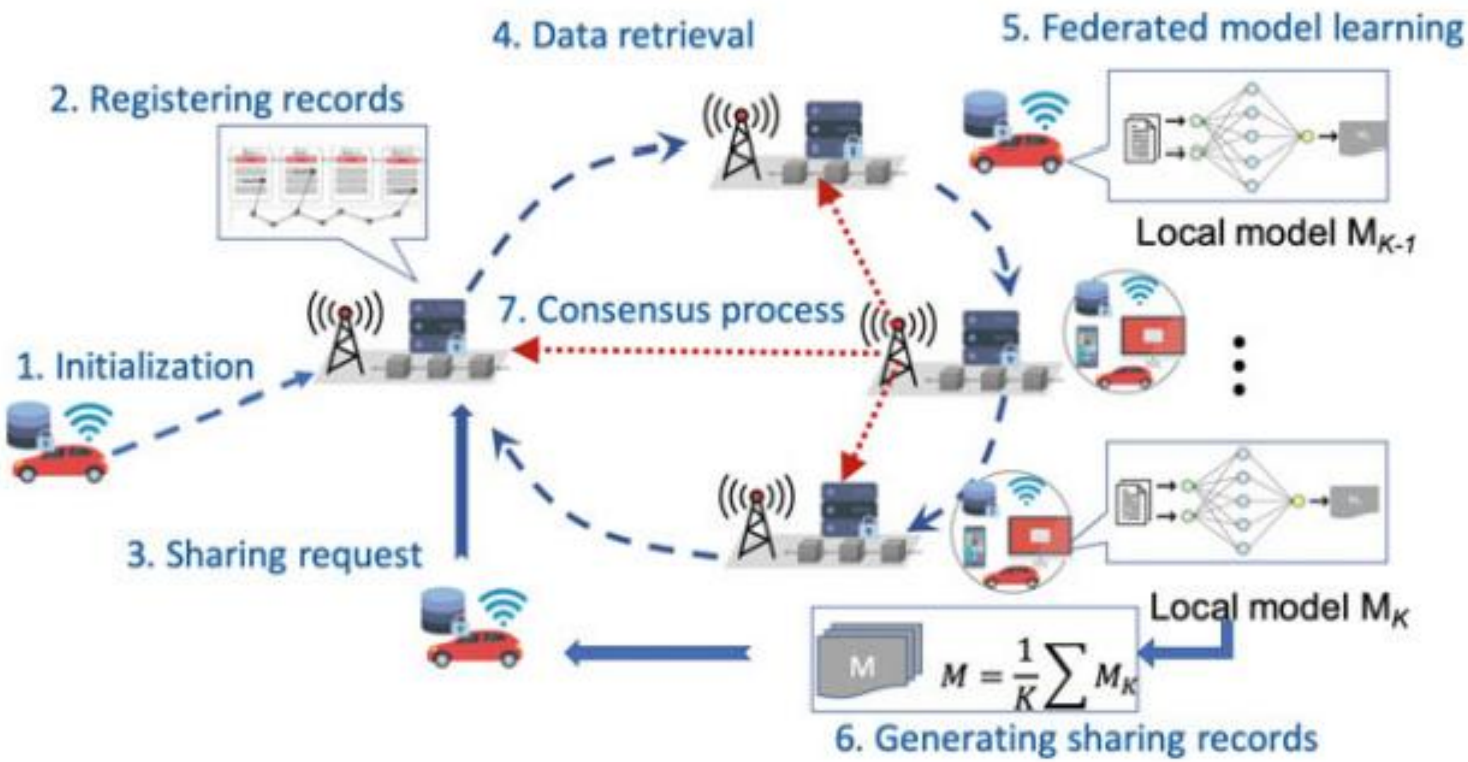
- `http://10.0.22.37:8000/containers/create/server?img_server=fed-server&port=5010&id=10&clients=4&algorithm=FedAvg&model=cnn&rounds=10&epochs=5&predict=true`

Rapid deployment of a testbed and run tests:

- `http://10.0.22.37:8000/run?img_server=fed-server&img_client=fed-client&model=cnn&clients=4&rounds=10&epochs=5&predict=true&predict_client=true`

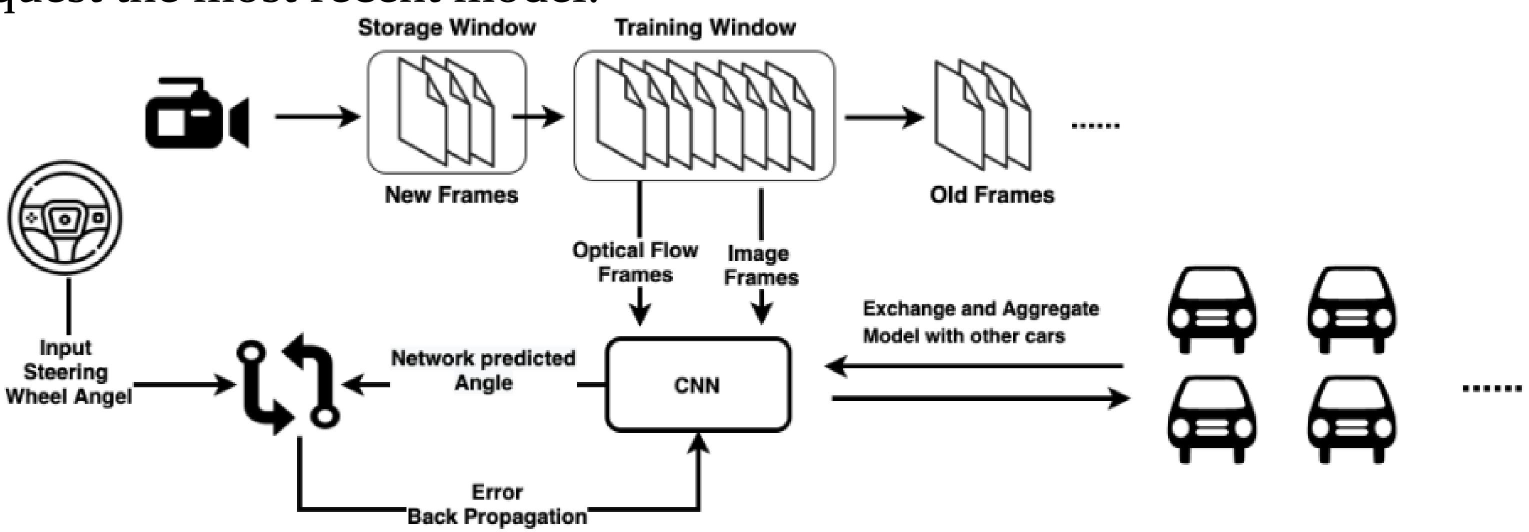
Edge-based Federated

- MEC-empowered model sharing
  - **Edge intelligence to wireless edge networks and enhances the connected intelligence among end devices in 6G networks.**



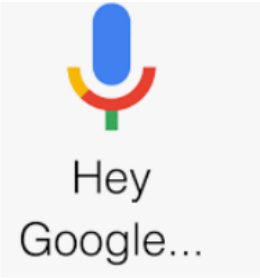
Federated learning in self-driving

- Edge vehicles compute the model locally; after completing each local training epoch, they retrieve the global model version and compare it to their local version.
- In order to form a global awareness of all local models, the central server performs aggregation based on the ratio determined by the global and local model versions.
- The aggregation server returns the aggregated result to the edge vehicles that request the most recent model.



Real Life examples

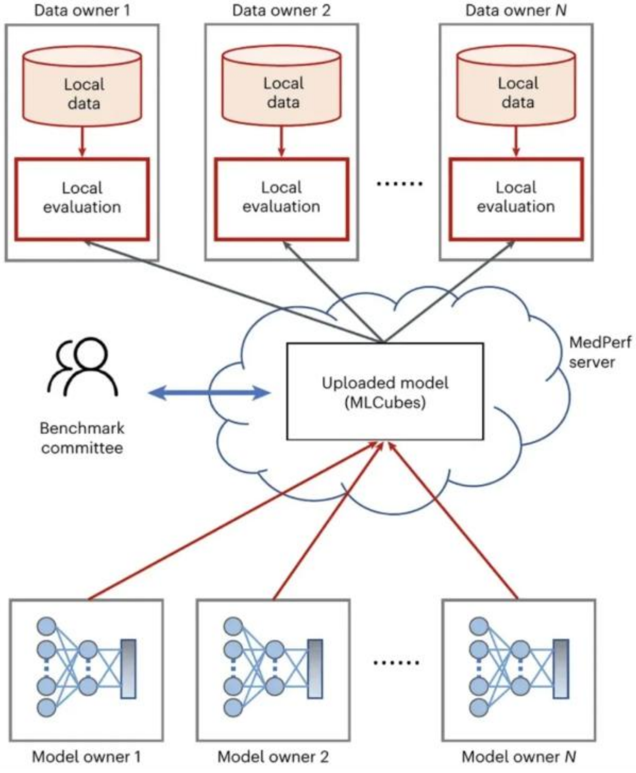
- **Mobile applications**
- Google employs federated learning to enhance on-device machine learning models, such as the “Hey Google” detection in Google Assistant, enabling users to issue voice commands. This approach allows the training of speech models directly on users’ devices without transmitting audio data to Google’s servers, thereby preserving user privacy.
- Federated learning facilitates the improvement of voice recognition capabilities by processing data locally, ensuring that personal audio information remains on the device.



https://research.aimultiple.com/federated-learning/ , post from 21st March 2025 ☺

Real Life examples

- **Healthcare**
- A growing push for federated learning in medical AI has led to initiatives like MedPerf, an open-source platform developed by a coalition of industry and academic partners.
- MedPerf focuses on federated evaluation of AI models, ensuring they perform effectively on diverse, real-world medical data while maintaining patient confidentiality. By combining technical innovations in federated learning with governance frameworks that establish clinically relevant benchmarks, these initiatives aim to drive the adoption of AI in healthcare without compromising trust or security.



https://research.aimultiple.com/federated-learning/ **, post from 21st March 2025 ☺**

Real Life examples

- **Autonomous vehicles**
- NVIDIA's AV Federated Learning platform, powered by NVIDIA FLARE, enables autonomous vehicle (AV) models to be trained collaboratively across different countries while preserving data privacy and complying with regional regulations like GDPR.
- Instead of centralized training, which can be costly and restricted by data transfer laws, federated learning allows models to be trained locally on country-specific data, **improving global model performance** without moving raw data.
- The platform integrates with existing machine learning systems and operates with a central server on AWS in Japan, supporting cross-border training. Since launch, it has produced over a dozen AV models, with performance matching or **exceeding locally trained** counterparts, and adoption has grown from 2 to 30 data scientists within a year.

https://research.aimultiple.com/federated-learning/ **, post from 21st March 2025 ☺**

Real Life examples

- **Predictive maintenance in smart manufacturing**
- As Industry 4.0 advances, AI-driven predictive maintenance helps manufacturers reduce downtime, extend equipment lifespan, and boost efficiency. However, its implementation faces challenges, including data privacy, security, and cross-border sharing restrictions.

- Federated learning addresses these issues by enabling manufacturers to develop predictive maintenance models without transferring sensitive industrial data. Instead of aggregating information from multiple plants or customers into a central repository, federated learning allows each site to train models locally. These models then contribute insights to a global predictive system without exposing proprietary data.

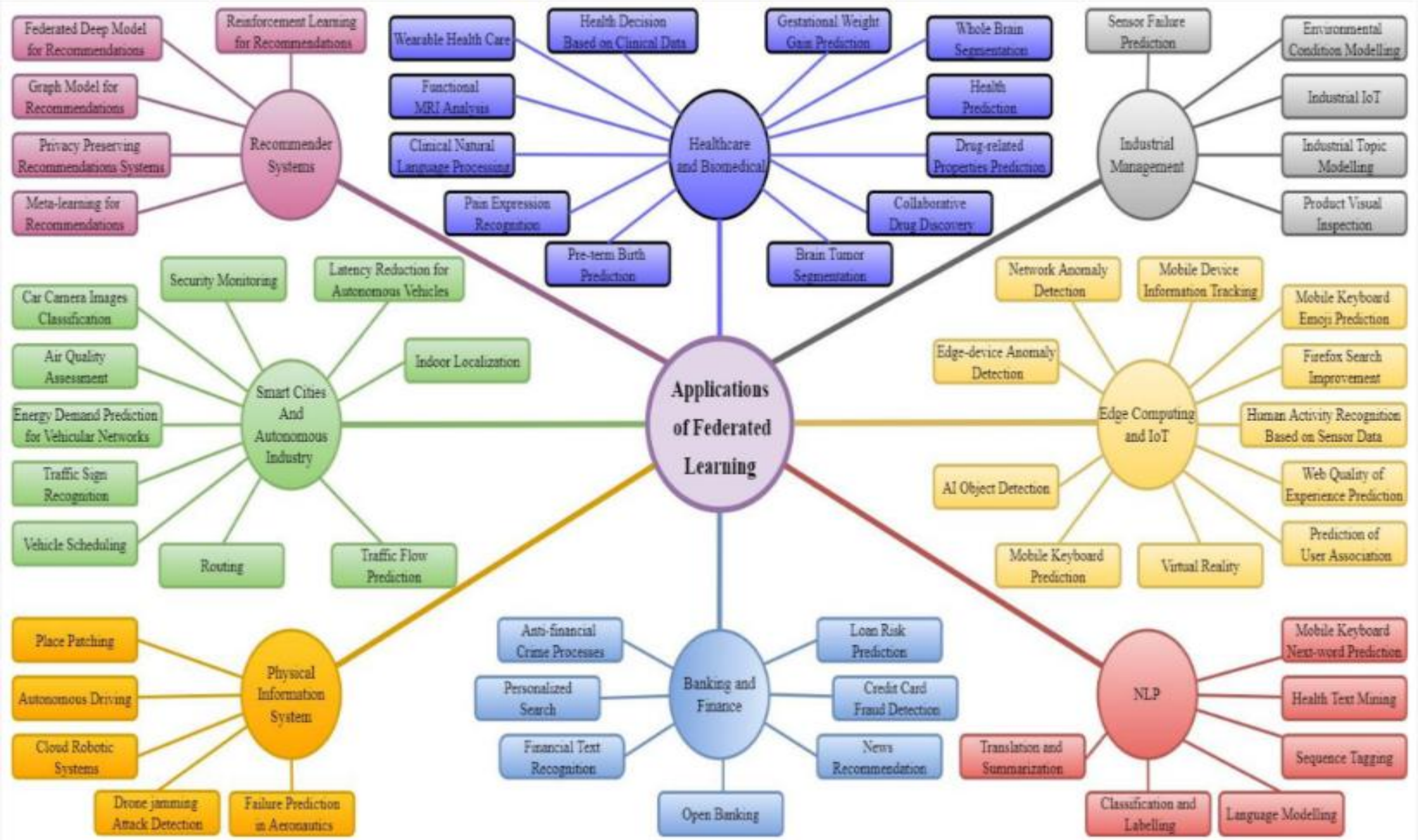
https://research.aimultiple.com/federated-learning/ , post from 21st March 2025 ☺

Real Life examples

- **Robotics and multi-robot navigation**
- Recent advancements in Deep Reinforcement Learning (DRL) have enhanced robotics by enabling automatic controller design, which is particularly important for swarm robotic systems. These systems require more sophisticated controllers than single-robot setups to achieve coordinated collective behavior.
- While DRL-based controller design has proven effective, its reliance on a central training server poses challenges in real-world environments with unstable or limited communication.
- Federated Learning (FL)-based DRL training strategy tailored for swarm robotics. Comparative evaluations under limited communication bandwidth demonstrate that FL-DRL offers increased generalization to diverse environments and real robots, whereas baseline methods struggle with bandwidth constraints.

https://research.aimultiple.com/federated-learning/ , post from 21st March 2025 ☺

Applications for Federated



Applications for Federated

Domain	Applications
Edge computing	FL is implemented in edge systems using the MEC (mobile edge computing) and DRL (deep reinforcement learning) frameworks for anomaly and intrusion detection.
Recommender systems	To learn the matrix, federated collaborative filter methods are built utilizing a stochastic gradient approach and secured matrix factorization using federated SGD.
NLP	FL is applied in next-word prediction in mobile keyboards by adopting the FedAvg algorithm to learn CIFG [93].
IoT	FL could be one way to handle data privacy concerns while still providing a reliable learning model
Mobile service	The predicting services are based on the training data coming from edge devices of the users, such as mobile devices.
Biomedical	The volume of biomedical data is continually increasing. However, due to privacy and regulatory considerations, the capacity to evaluate these data is limited. By collectively building a global model for the prediction of brain age, the FL paradigm in the neuroimaging domain works effectively.
Healthcare	Owkin [31] and Intel [32] are researching how FL could be leveraged to protect patients' data privacy while also using the data for better diagnosis.
Autonomous industry	Another important reason to use FL is that it can potentially minimize latency. Federated learning may enable autonomous vehicles to behave more quickly and correctly, minimizing accidents and increasing safety. Furthermore, it can be used to predict traffic flow.
Banking and finance	The FL is applied in open banking and in finance for anti-financial crime processes, loan risk prediction, and the detection of financial crimes.

- https://www.pdl.cmu.edu/SDI/2019/slides/2019-09-05Federated%20Learning.pdf
- https://wires.onlinelibrary.wiley.com/doi/epdf/10.1002/widm.1443
- https://medium.com/tensorflow/introducing-tensorflow-federated-a4147aa20041