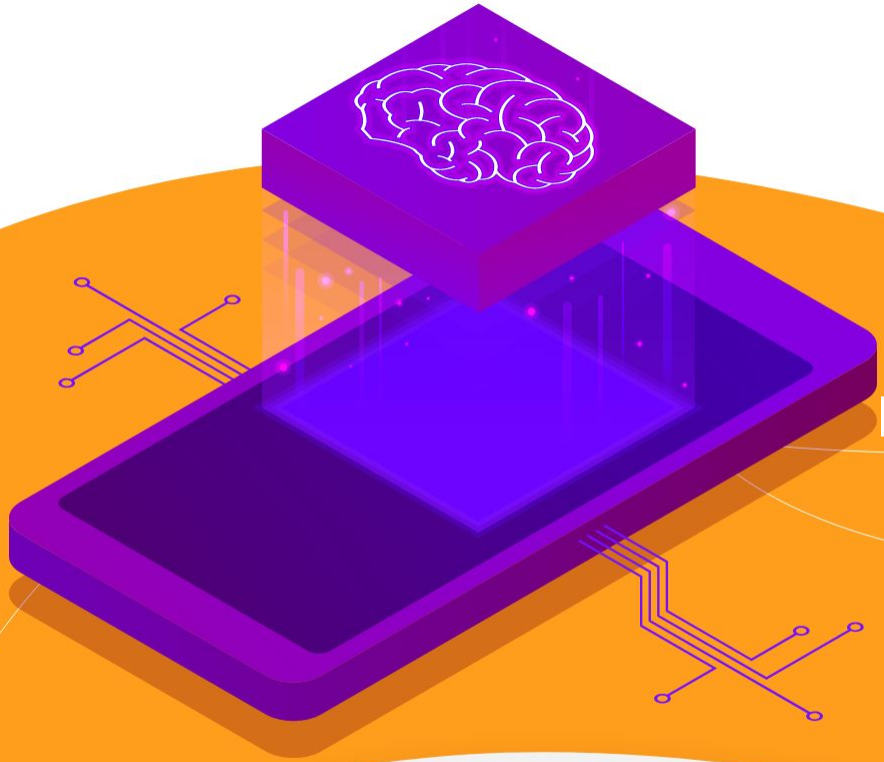


H.IAAC | HUB DE INTELIGÊNCIA ARTIFICIAL E ARQUITETURAS COGNITIVAS

Hands-on em Federated Learning usando Flower

Apresentadores: Alexandre F. S. Osorio, Adson Nogueira Alves; Maria Vitória R. Oliveira

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÕES | PÁTRIA AMADA BRASIL GOVERNO FEDERAL | Softex | ELDORADO INSPIRAÇÃO PELO NOVO | UNICAMP | Instituto de Computação UNIVERSIDADE ESTADUAL DE CAMPINAS | FEEC

O HUB

- O Hub para Inteligência Artificial e Arquiteturas Cognitivas (H.IAAC) visa expandir os limites da IA para permitir mais do que a tomada de decisão local em cenários muito controlados
- Desafios em várias dimensões

Modelos de aprendizado em agentes cognitivos

Aprendizado e execução do agente distribuído

Arquitetura cognitiva mobile com interação externa

Representação de conhecimento multimodal

Processamento de Linguagem Natural

O Projeto Arquiteturas Cognitivas

- Por iniciativa do Ministério da Ciência, Tecnologia e Inovações (MCTI), com coordenação da Softex, e execução do Instituto ELDORADO e da Unicamp, o Hub de Inteligência Artificial e Arquitetura Cognitiva tem o objetivo de desenvolver e disseminar conhecimento sobre tecnologias capazes de integrar diversos recursos de inteligência em dispositivos móveis, tornando-os hábeis em tomar decisões.
- O Instituto de Computação (IC) e a Faculdade de Engenharia Elétrica e de Computação (FEEC) da Unicamp junto ao ELDORADO farão parte da execução do projeto, enquanto a Softex ficará responsável pela gestão e manejo dos recursos advindos da Lei de Informática.
- Projeto conta com um aporte do MCTI no valor de R$ 7,6 milhões.

AGENDA

- Introdução ao aprendizado distribuído
- Introdução ao framework Flower
- Tutorial: implementação de aprendizado distribuído de um modelo de classificação
  - Configuração do ambiente para o experimento
  - Apresentação do código em Python
  - Execução do experimento

EQUIPE UNICAMP

COORDENADOR

Leandro A. Villas

PESQUISADOR ASSOCIADO

Luiz Bittencourt

PESQUISADOR COLABORADOR

Antonio A. F. Loureiro

PÓS-DOUTORANDO

Allan M. de Souza

DOUTORANDO

Adson Alves

MESTRANDO

Gabriel Campos

DOUTORANDA

Maria Vitória

GRADUNDO

Gabriel Teston

GRADUNDO

Gabriel Talasso

EQUIPE ELDORADO

LÍDER DE PROJETO
Fábio Grassioto

GERENTE DE PROJETO
Michelle Scarassati

PESQUISADORA
Amparo Muñoz

PESQUISADOR
Alexandre Osorio

PESQUISADOR
Daniel Miranda

PESQUISADOR
Felipe de Paula

PESQUISADOR
Sildolfo Gomes

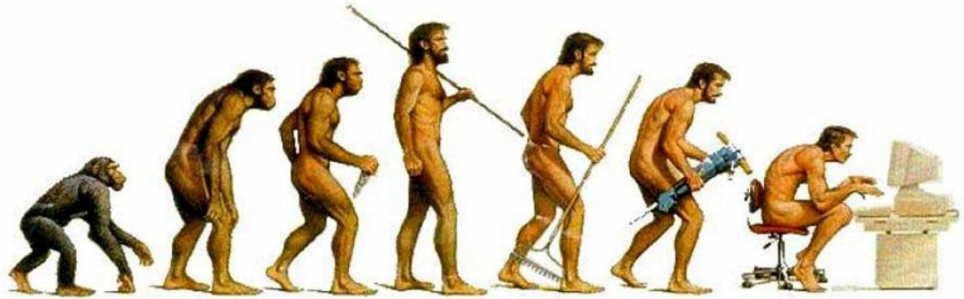
PESQUISADOR
Wandemberg Gibaut

Introdução

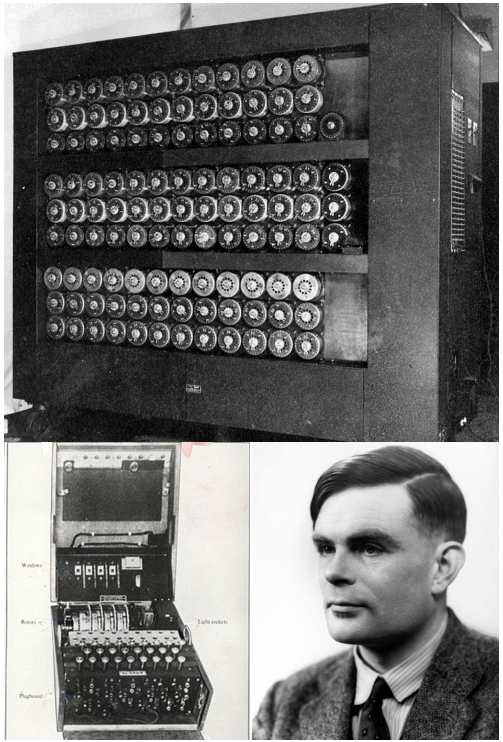
Adson Nogueira Alves

Machine Learning - Introdução & Motivação

"We share information, we create and pass on knowledge. That's the means by which humans are able to adjust to new situations, and it's what differentiates humans from our earlier ancestors, and our earlier ancestors from primates ".
[Massey, 2013]



Machine Learning - Introdução & Motivação

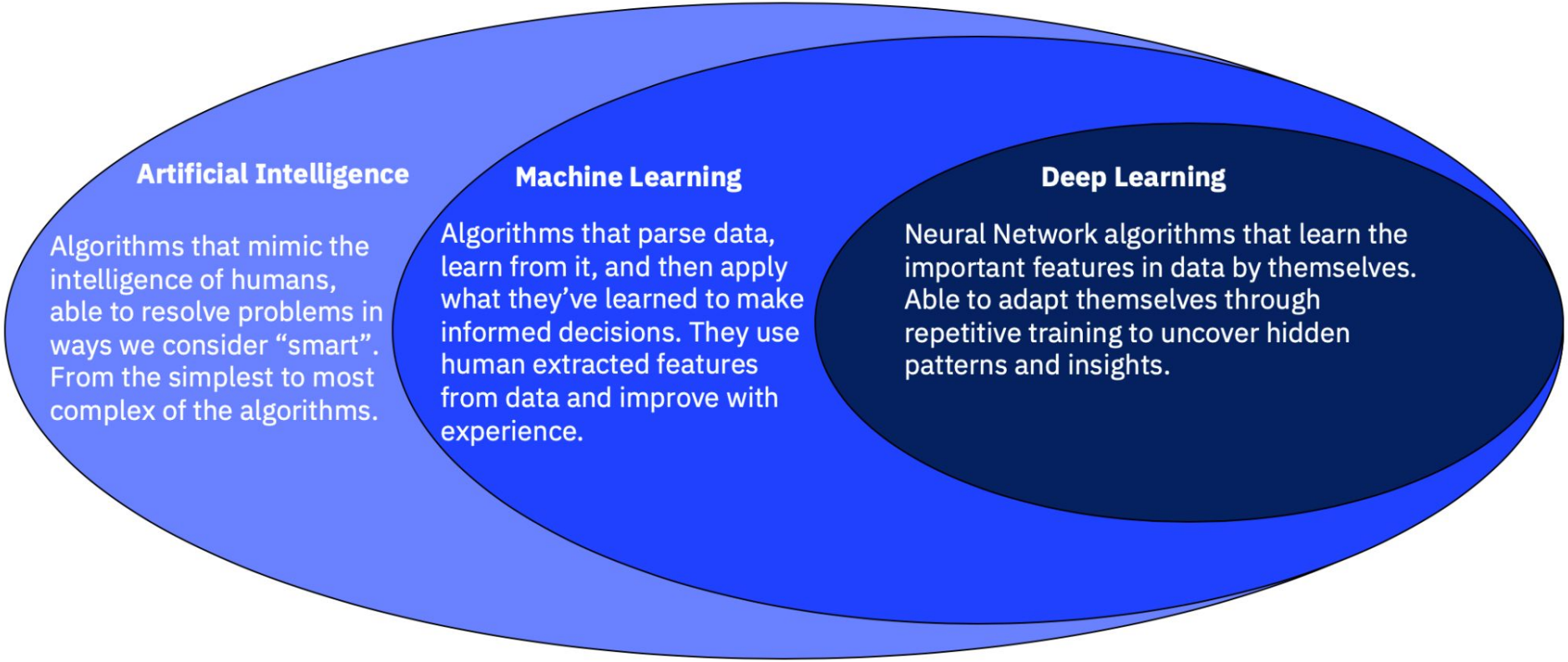


Among the many possible ways to de ne AI, Raymond Kurzweil said that AI is:

"The art of creating machines that perform functions that require intelligence when performed by people ".

[Kurzweil, 1990]

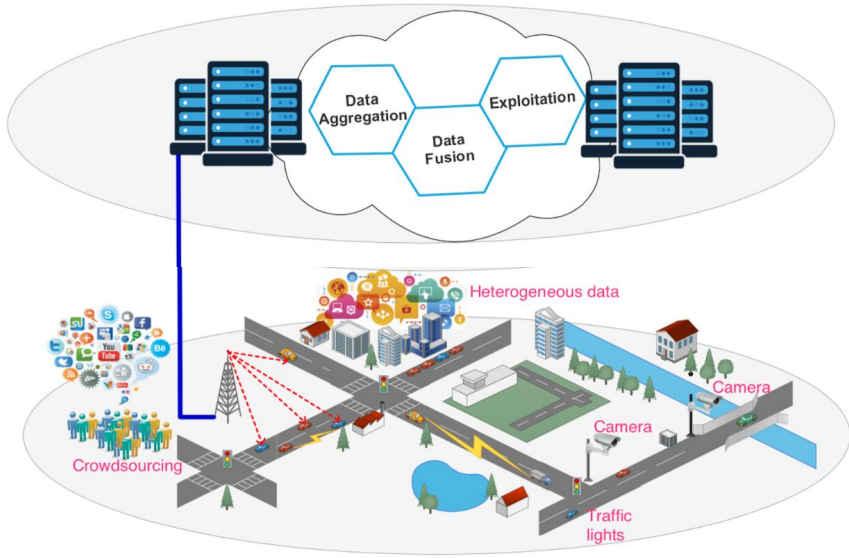
Machine Learning - Introdução & Motivação



Fonte: IBM

Machine Learning - Introdução & Motivação

- *Advento dos dispositivos massivamente conectados, avanços da **inteligência artificial**, abriram caminho para o desenvolvimento de **sistemas inteligentes***
- *Baseiam-se na coleta de dados dos dispositivos*
- *Aplicação de algoritmos de aprendizado de máquina*
- *Geração de conhecimento a partir de dados para projetar soluções inteligentes nas mais diversas áreas.*

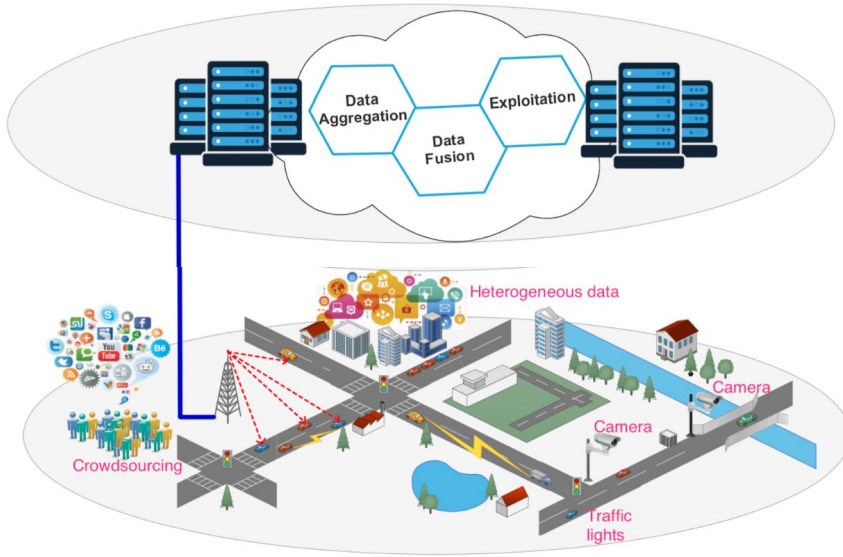


Fonte: IBM

Machine Learning - Introdução & Motivação

Problemas:

- Inteligência desses serviços/soluções não estarem presentes nos dispositivos onde os dados são gerados
- *Aumento da comunicação*
- *Privacidade dos dados*
- *...*

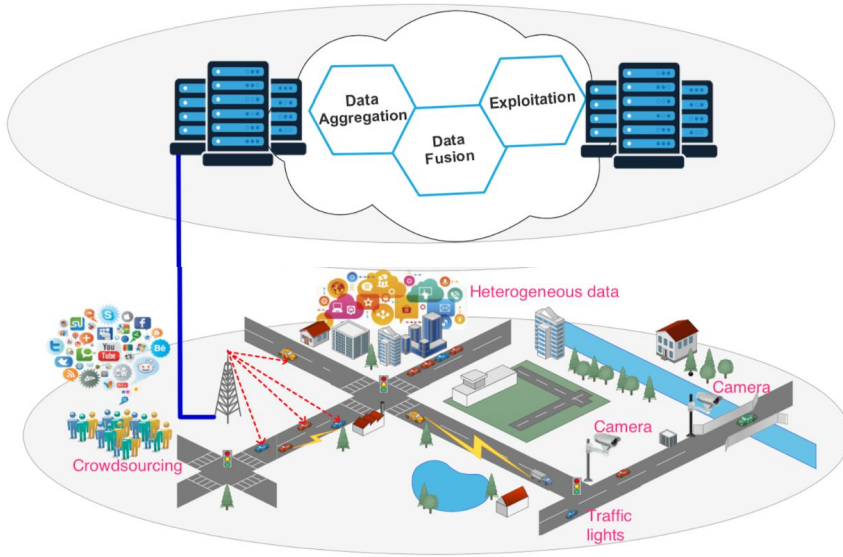


Fonte: IBM

Machine Learning - Introdução & Motivação

Problemas:

- Inteligência desses serviços/soluções não estarem presentes nos dispositivos onde os dados são gerados
- *Aumento da comunicação*
- *Privacidade dos dados*
- *...*



Como reduzir esses problemas mantendo a eficiência de um sistema inteligente?

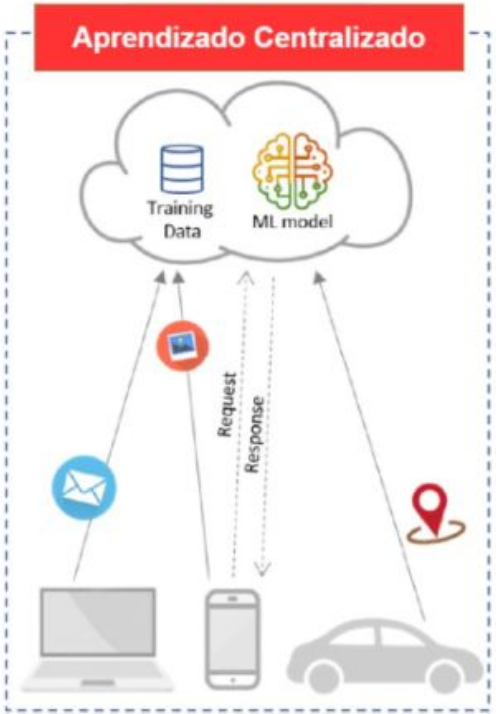
ARQUITETURAS DE APRENDIZADO PARA SISTEMAS INTELIGENTES

ARQUITETURAS DE APRENDIZADO PARA SISTEMAS INTELIGENTES

- **Visão geral:** *Dispositivos finais enviam seus **dados** para um servidor onde os dados são utilizados para treinamento de um modelo global*

- **Vantagem:** *Modelos eficientes e robustos pelo fato do servidor ter acesso aos dados de todos os clientes*

- **Desvantagem:** *Problemas relacionados a privacidade dos dados dos usuários*



ARQUITETURAS DE APRENDIZADO PARA SISTEMAS INTELIGENTES

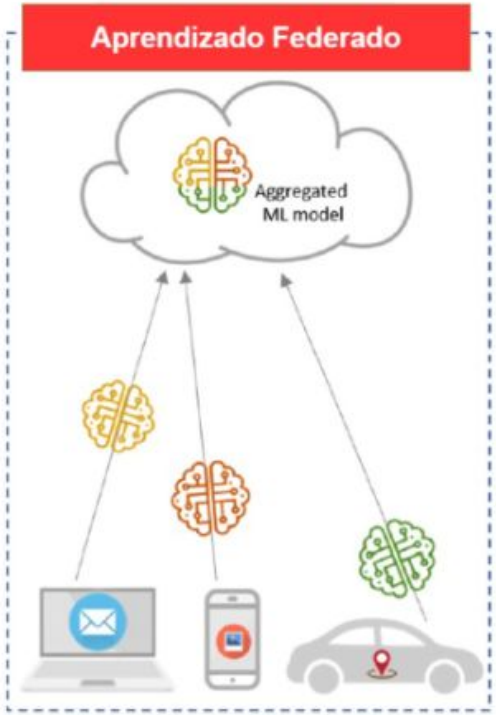
- **Visão geral:** *Dispositivos finais treinam modelos locais com seus próprios dados sem o compartilhamento de informações com o servidor ou outros dispositivos*
- **Vantagem:** *Garante a privacidade dos dados, pois nenhum informação é compartilhada*
- **Desvantagem:** *Modelos menos eficientes devido aos dados limitados de cada usuário; generalização do problema é limitada*

ARQUITETURAS DE APRENDIZADO PARA SISTEMAS INTELIGENTES

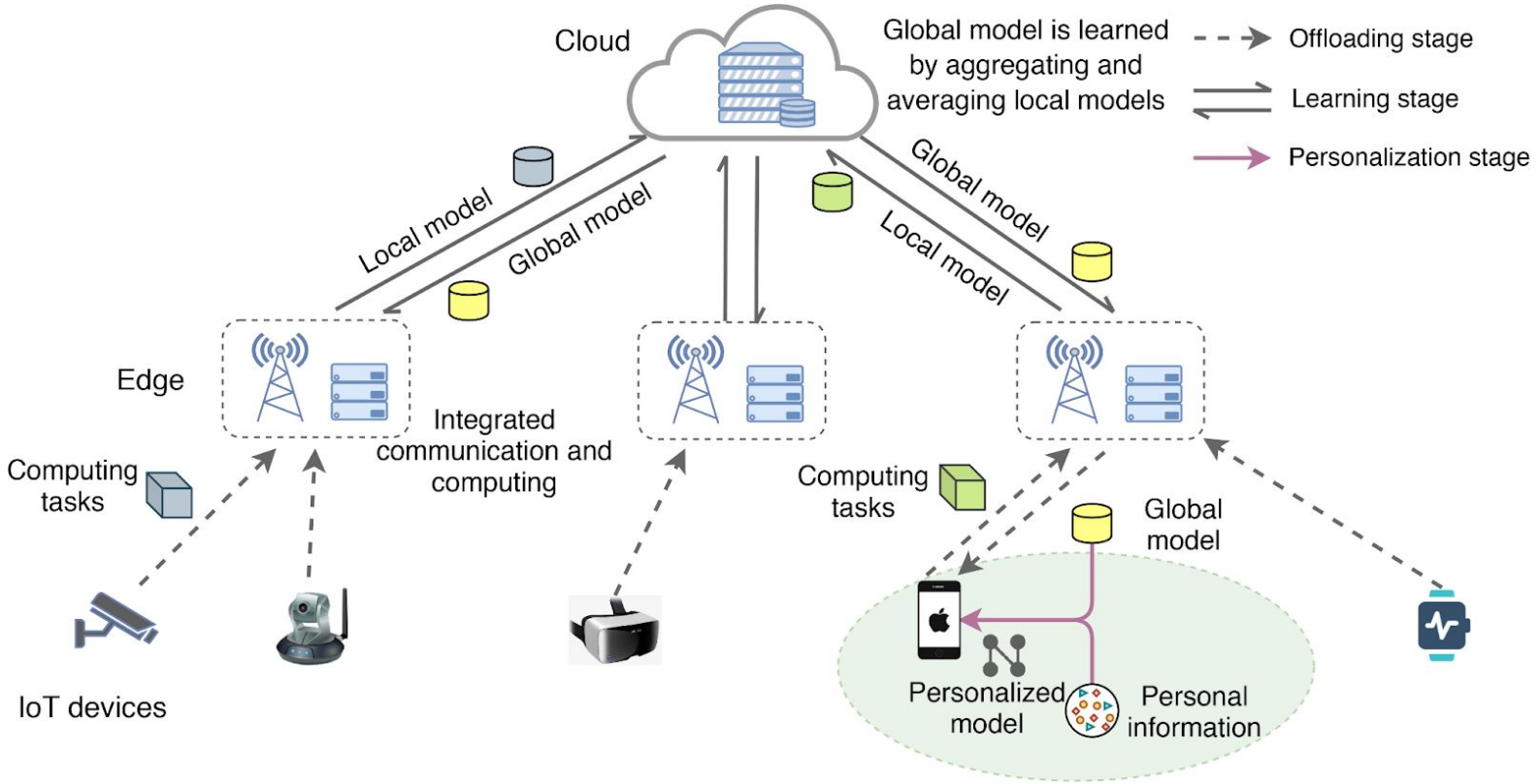
- ***Visão geral:*** *Dispositivos finais fazem o treinamento de um modelo com seus dados locais, em seguida, os modelos treinados são compartilhados com um servidor, que realiza a agregação e atualização dos modelos dos dispositivos finais*

- ***Vantagem:*** *Garante a privacidade e mantém uma performance eficiente devido ao compartilhamento das experiências de cada dispositivo*

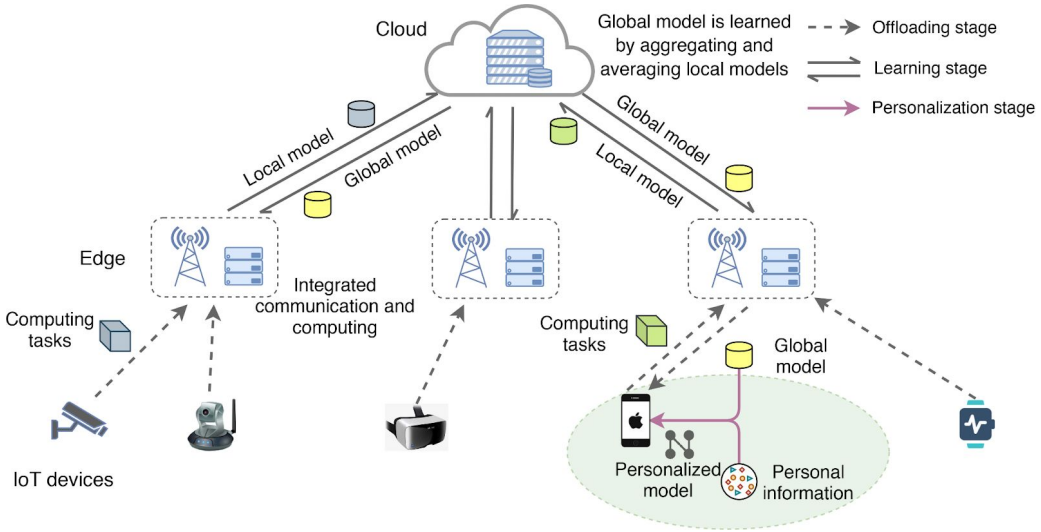
- ***Desvantagem:*** *Necessita de algoritmos eficientes para compartilhamento e agregação de modelos*



AGREGAÇÃO DE MODELOS PARA FL (LOCAL E FEDERADO)

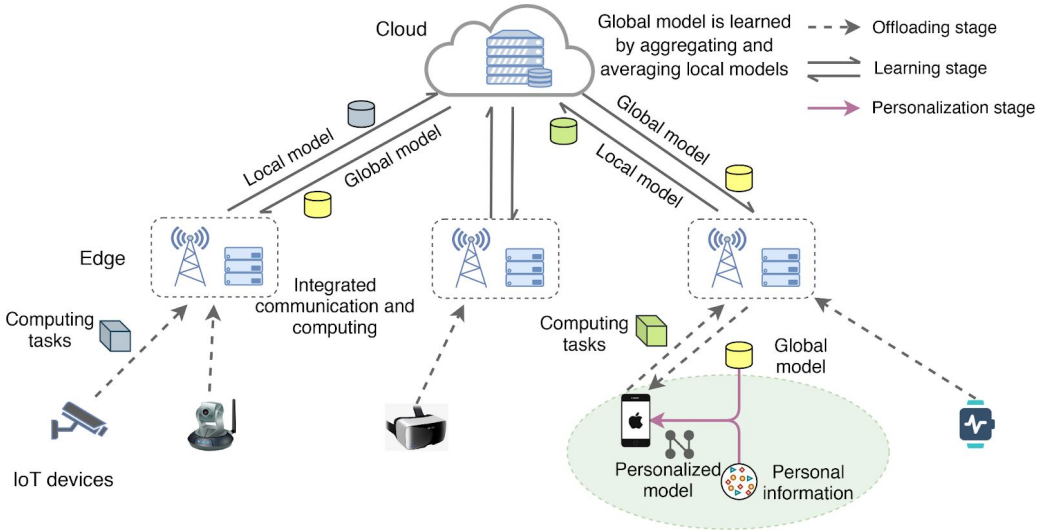


AGREGAÇÃO DE MODELOS PARA FL (LOCAL E FEDERADO) - VANTAGENS



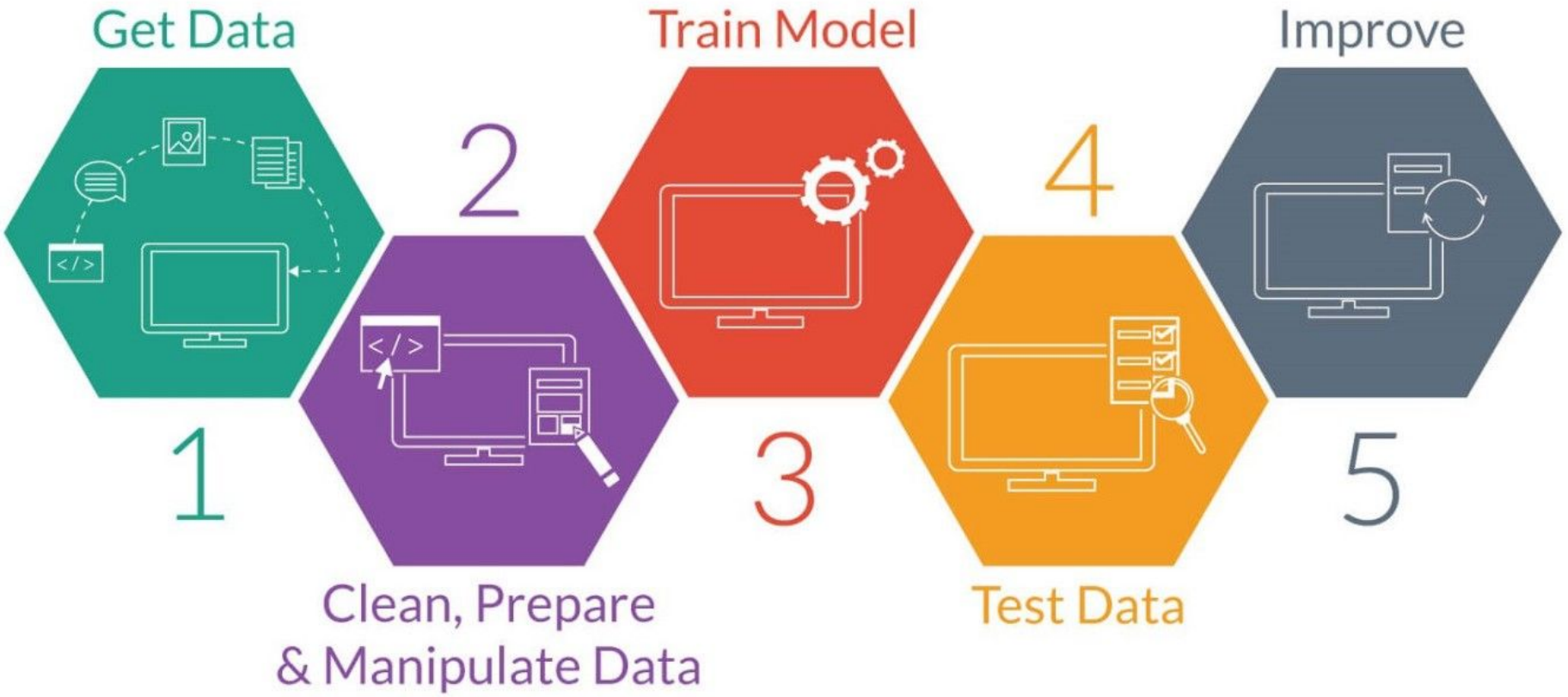
- Privacidade de dados
- Etapa fundamental do aprendizado federado
- Generalização dos modelos
- Trabalhar com modelos heterogêneos
- Modelo global compartilhado
- Herança de aprendizado
- Pluralidade de hardwares
- Robustez do modelo aprendido

AGREGAÇÃO DE MODELOS PARA FL (LOCAL E FEDERADO) - DESAFIOS



- Non-IID (Distribuído de forma não independente e idêntica)
- Custo de comunicação
- Comunicação limitada
- Dispositivos offline
- Modelos retardatários / obsoletos
- Múltiplas etapas local gerando *biases*
- Tempo de convergência para modelos heterogêneos

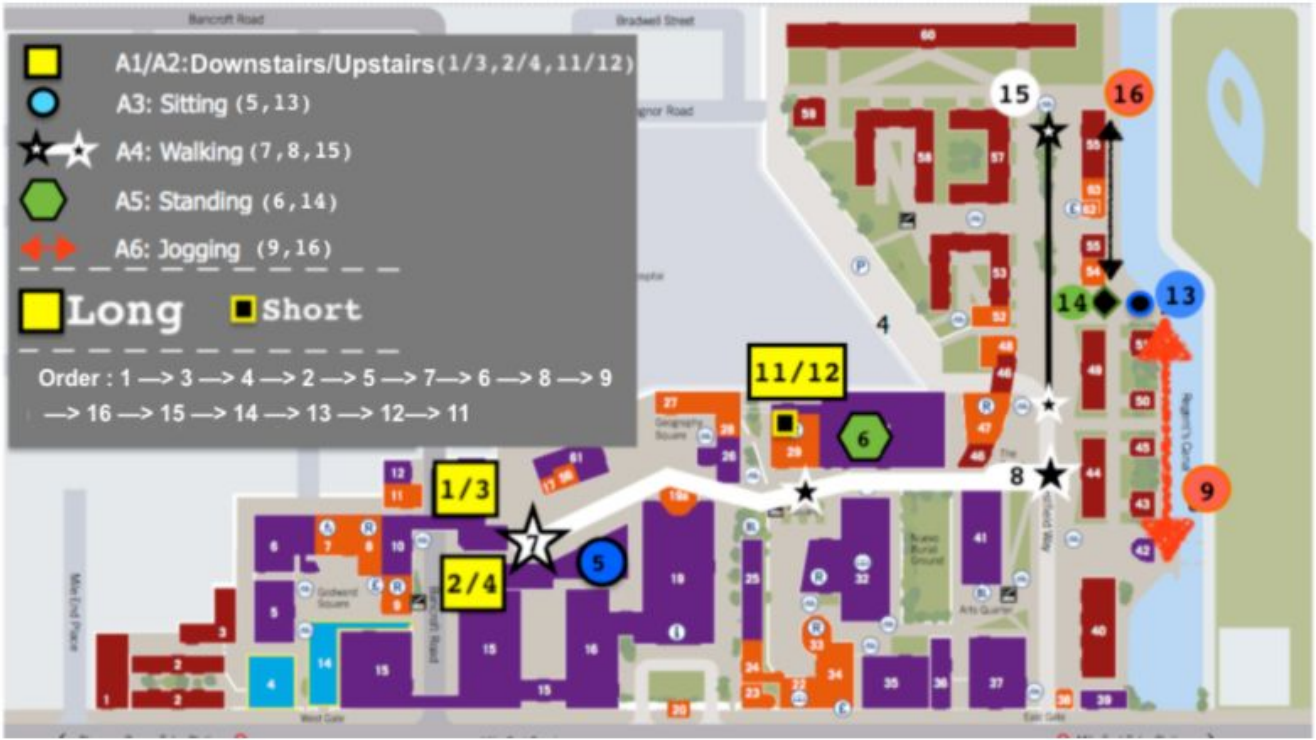
Workflow of a ML Project



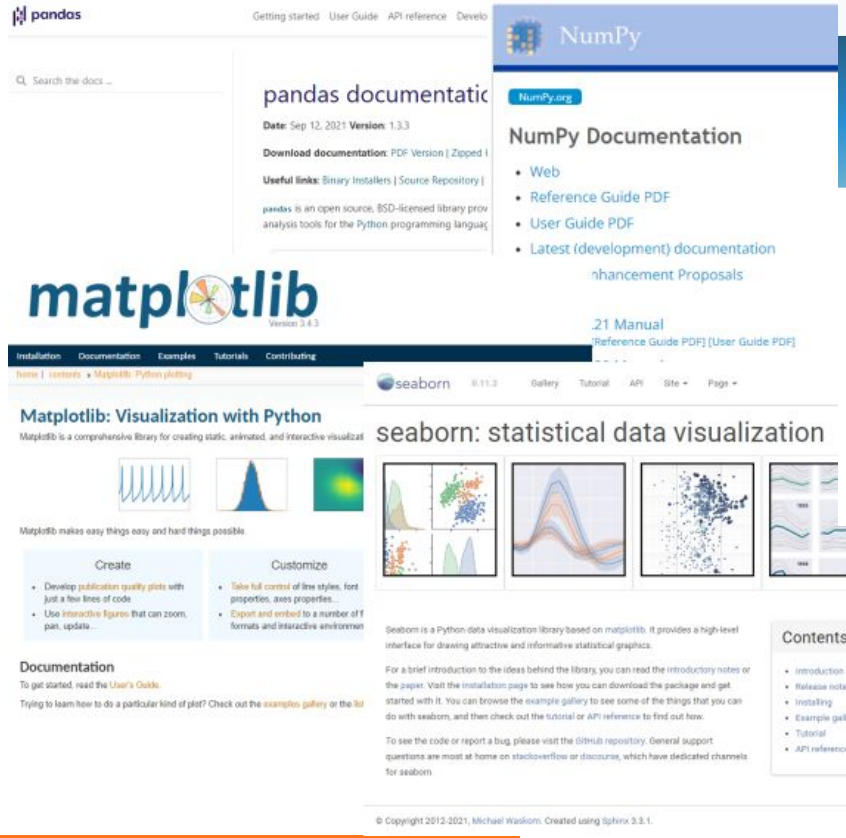
Pré Processamento de Dados

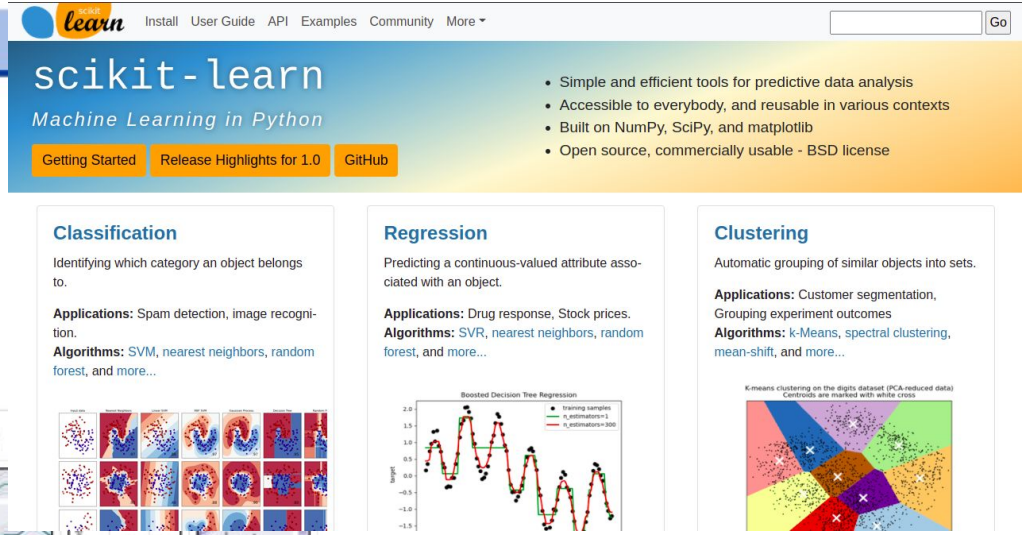
Adson Nogueira Alves

Dataset - Motion Sense

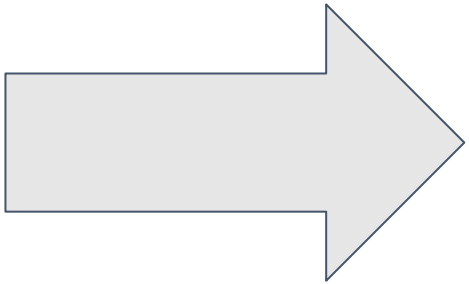


Libraries / Frameworks





Notebook - Colab



colab

https://colab.research.google.com/drive/1N-pl-snsDgzk-cCuPYV_hXsPB6t1SwxE?usp=sharing

Aprendizado Centralizado

Maria Vitória

Aprendizado Centralizado

No aprendizado centralizado, realiza-se a transmissão contínua de dados para a nuvem. Sendo assim, nos servidores de alto desempenho, realiza-se a análise dos dados, extração de recursos e treinamentos de modelos. Quanto mais interações com serviços disponíveis na nuvem, mais dados de treinamento são coletados e, portanto, aplicativos mais inteligentes são produzidos.

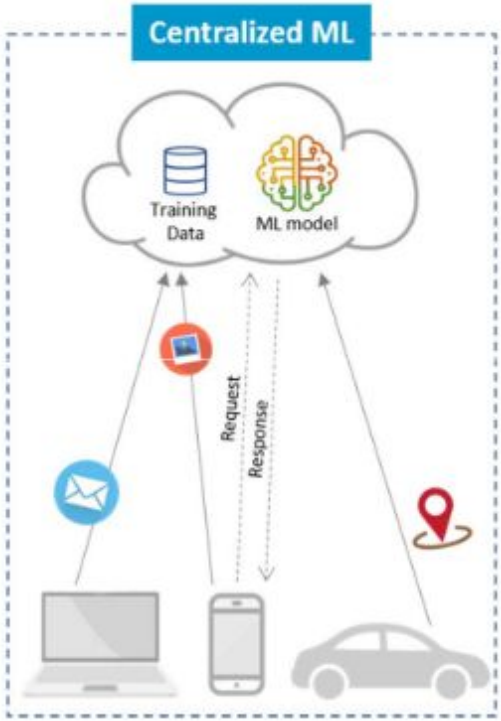
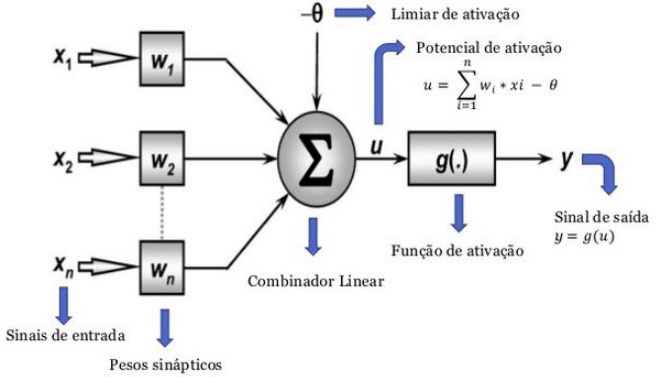
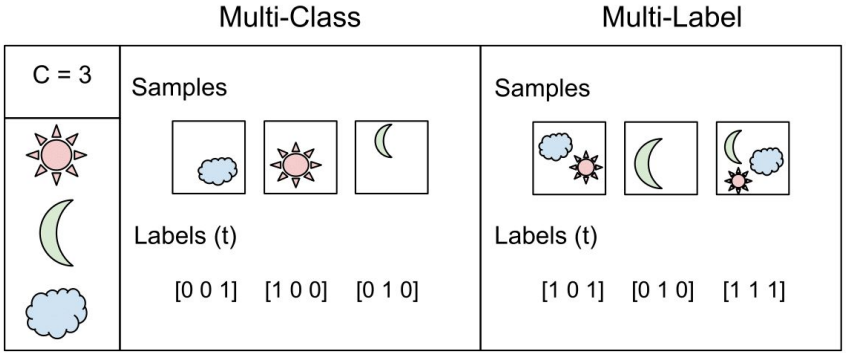


Figura 1 - Aprendizado centralizado. [1]

Aprendizado supervisionado

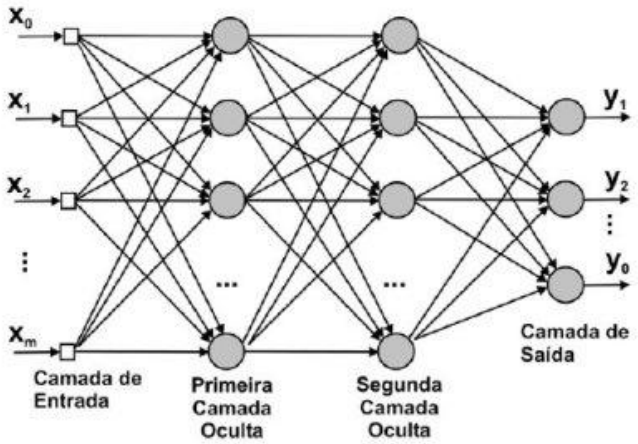
- Cada amostra de treinamento é composta pelos sinais de entradas e suas saídas;
- Os pesos sinápticos e o limiar são inicializados aleatoriamente;
  - Durante o treinamento os pesos são continuamente ajustados;





Redes Neurais Artificiais

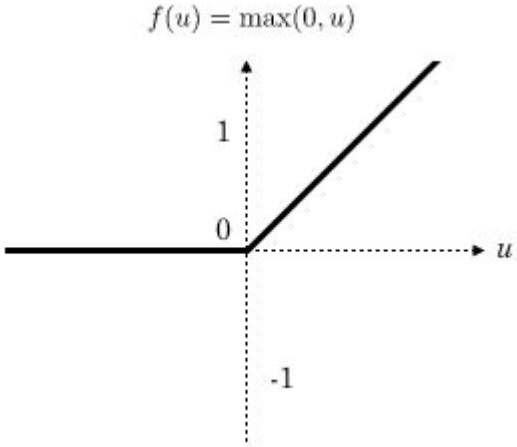
- Rede Neural de camadas densas;
- Modelo Sequencial: pilha linear de camadas;
- Número de entradas: features;
- Neurônios da última camada: classes.



Função de Ativação

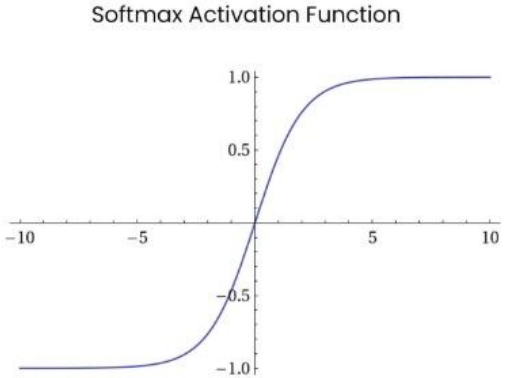
- Rectified Linear Activation Unit (ReLU):

$$f(x) = max(0, x)$$



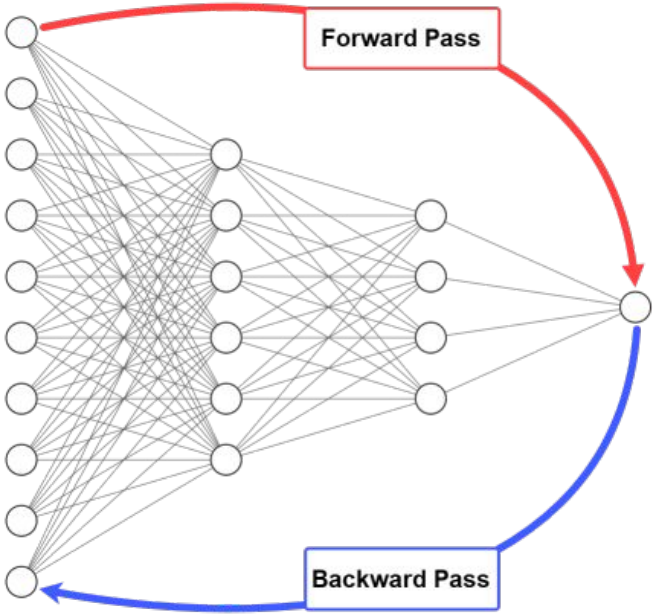
- Softmax: usada em redes neurais de classificação. A saída de uma rede neural representa a probabilidade dos dados serem de uma das classes definidas.

$$softmax(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$



Backpropagation

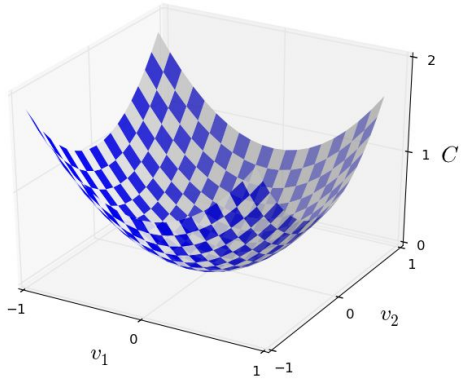
- O termo backpropagation define a forma com que a rede é treinada;
- Com o erro calculado, o algoritmo corrige os pesos em todas as camadas, partindo da saída até a entrada;
- A função que calcula o erro precisa ser diferenciável, assim como a função de ativação.
- Através da derivada da função do erro, pode-se encontrar os pesos que minimizam o erro;
- Um método comum é a descida do gradiente;

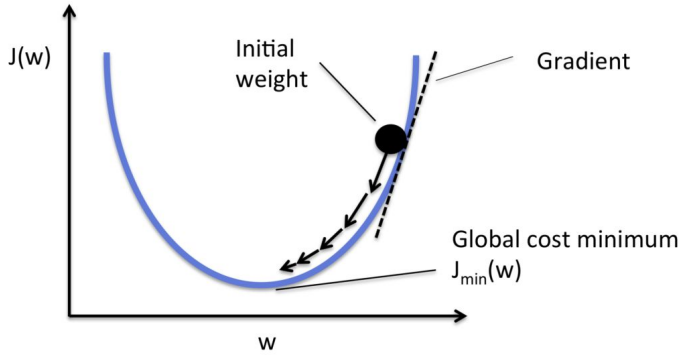


Descida do gradiente

Descida do Gradiente é um algoritmo de otimização usado para encontrar os valores de parâmetros (pesos e limiares) de uma função que minimizam uma função de custo [3].

- Gradiente descendente;
- Gradiente descente estocástico;





Keras

- API de aprendizado profundo;
- Permite a experimentação rápida;
- Executada na plataforma de aprendizado de máquina TensorFlow;
  - O TensorFlow é uma plataforma de aprendizado de máquina de código aberto;
- Principais estruturas de dados: camadas e modelos;

Aprendizado Centralizado

O principal obstáculo dos métodos de aprendizado de máquina (ML) tradicionais mudou de ser capaz de realizar inferência a partir de pequenas quantidades de dados de treinamento para como lidar com conjuntos de dados de treinamento com grande escala e altas dimensões.

- Falta de memória;
- Tempo de treinamento inviável;
- Violação de privacidade;

Figura 1 - Aprendizado centralizado. [1]

Aprendizado distribuído

Aprendizado Distribuído

Algoritmos de aprendizado distribuído são projetados para alcançar pelo menos um dos objetivos a seguir:

- Escalabilidade:
  - Paralelismo de dados;
  - Paralelismo de modelo;
  - Paralelismo de tarefas;
- Preservar a privacidade:
  - Várias entidades e cada uma contém um subconjunto de dados de treinamento;

Figura 2 - Aprendizado distribuído. [1]

Aprendizado Federado

O Aprendizado Federado (Federated Learning - FL) é uma técnica que permite que dispositivos de borda aprendam de forma colaborativa um modelo de previsão compartilhado, mantendo seus dados de treinamento no dispositivo, dissociando a capacidade de fazer aprendizado de máquina da necessidade de armazenar os dados em nuvem.

- A heterogeneidade de sistemas e escalabilidade dificultam a implementação do FL.
- Muitos trabalhos de simulação de algoritmos de FL não suportam o estudo de cargas de trabalho de FL escaláveis em dispositivos de borda heterogêneos.

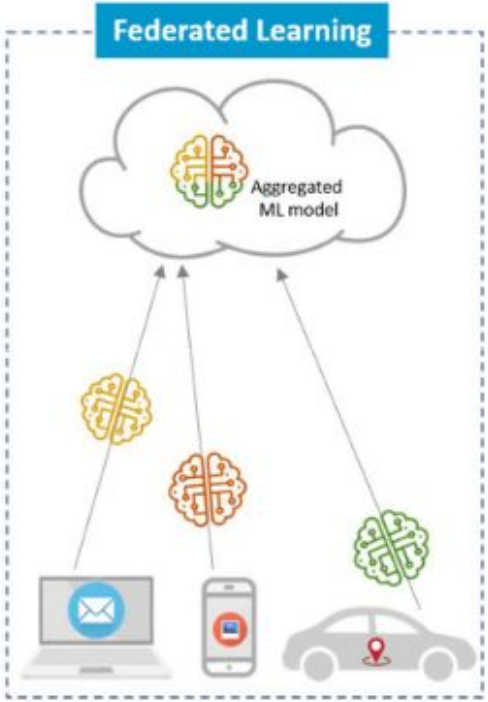
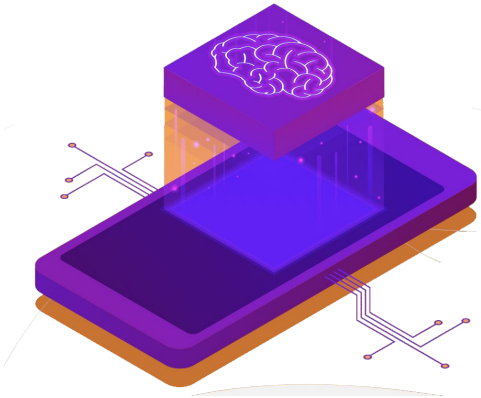


Figura 3 - Aprendizado federado. [1]

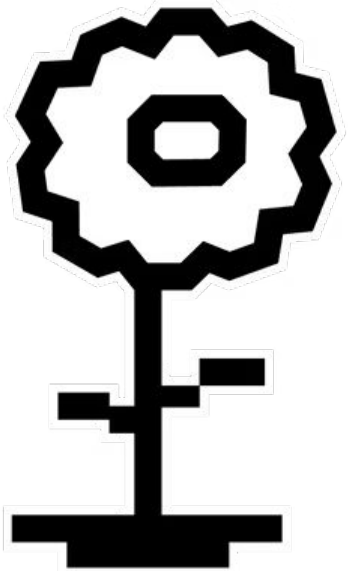


FLOWER

Um framework de código aberto para federated learning

Flower

- Pode ser usado com qualquer estrutura de aprendizado de máquina: PyTorch, TensorFlow, Hugging Face Transformers, PyTorch Lightning, MXNet, scikit-learn, TFLite;
- Cliente-agnóstico: interoperável com diferentes linguagens de programação, sistemas operacionais e hardware.
- Agnóstico à comunicação: permite diferentes abordagens de comunicação
- Arquitetura voltada para escalabilidade



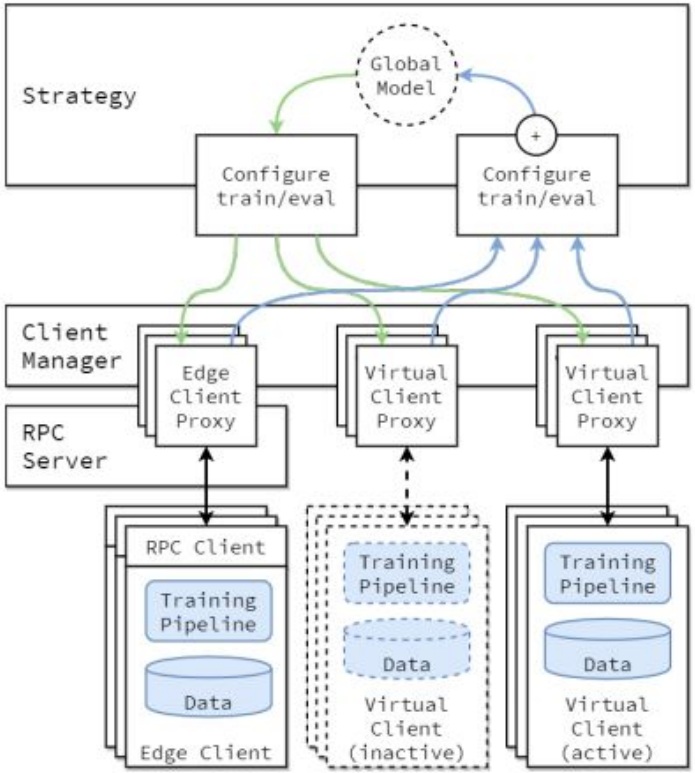
Flower core

Lado do servidor

- **Client Manager**: gerencia um conjunto de objetos ClientProxy, cada um representando um único cliente conectado ao servidor, que são responsáveis por enviar e receber mensagens do Protocolo Flower de e para o cliente real.
- **FL loop**: orquestra o processo de aprendizagem: pede ao Strategy para configurar a próxima rodada do FL, envia essas configurações para os clientes, recebe as atualizações (ou falhas) do cliente e delega a agregação de resultados ao Strategy.
- **Strategy**: abstração que permite a customização da lógica para seleção de clientes, a agregação de parâmetros e a avaliação do modelo federado ou centralizado

Lado do cliente

- Espera por mensagens do servidor.
- Reage às mensagens recebidas chamando as funções de treinamento e avaliação fornecidas pelo usuário.



Flower core

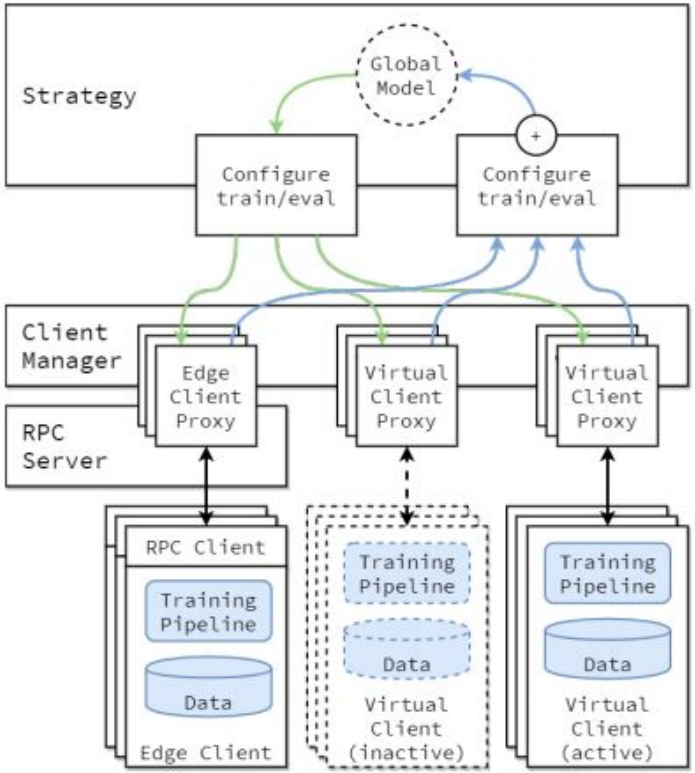
Virtual Client Engine

Agenda, instancia e executa os clientes de forma transparente para o usuário e o servidor.

- simplifica a paralelização de trabalhos
- garante que o hardware não seja subutilizado
- permite portar o mesmo experimento para uma ampla variedade de configurações, sem reconfiguração
- permite a execução de cargas de trabalho de FL de forma escalável

Edge Client Engine

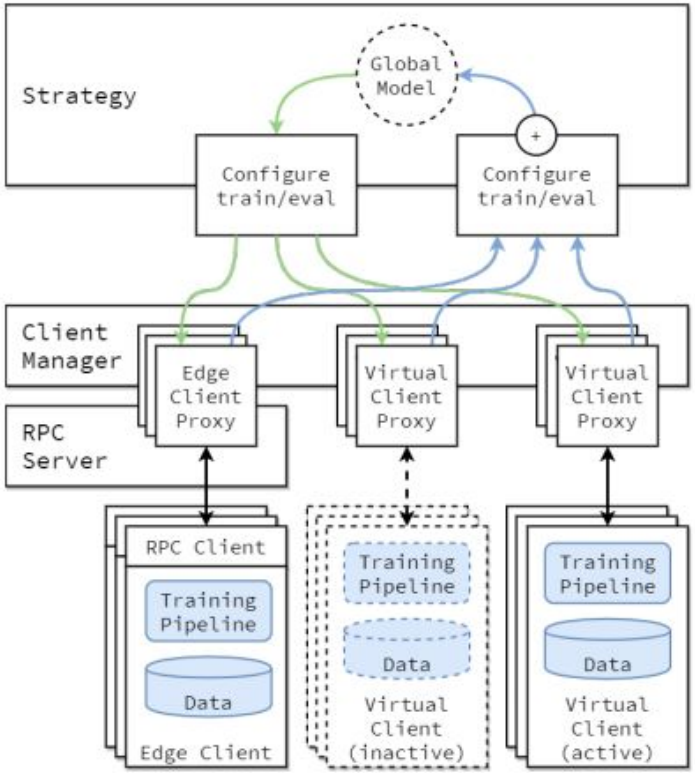
Dispositivos como smartphones podem exigir uma pilha de software mais restrita para executar cargas de trabalho de ML. Para contornar essa limitação, o Flower fornece uma integração de baixo nível manipulando diretamente as mensagens do protocolo no cliente.



Flower core

O servidor não tem conhecimento da natureza dos clientes conectados, o que permite treinar modelos em plataformas e implementações de clientes heterogêneos.

A estrutura gerencia também a conexão, o ciclo de vida do cliente, os tempos limite e a manipulação de erros.



Flower usa gRPC entre *server* e *clients*

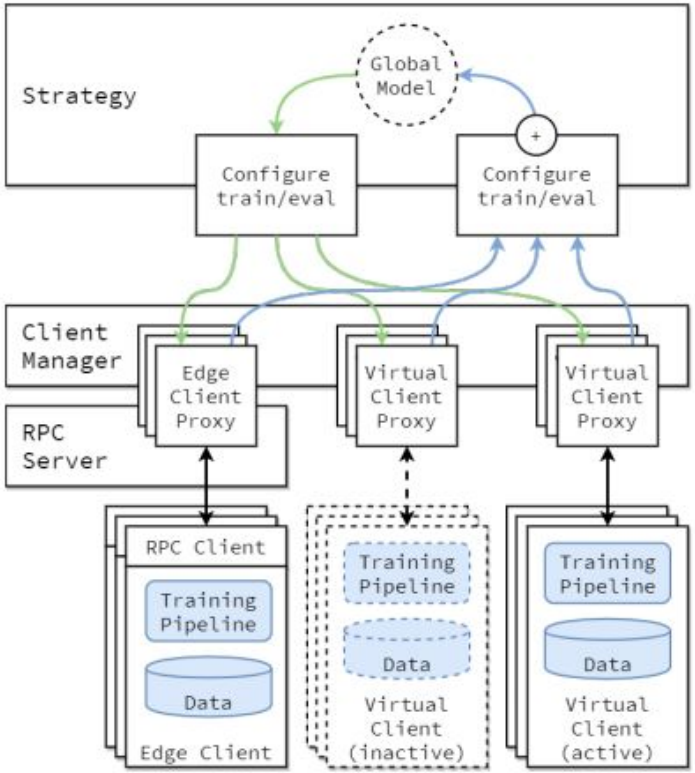
gRPC (Google Remote Procedure Call) usa

- HTTP/2 para transporte
- Protocol Buffers como a linguagem de descrição da interface

Fornece recursos como

- Autenticação
- *Streaming* bidirecional e controle de fluxo
- *Bindings* bloqueantes ou não
- Timeouts

Disponível em várias linguagens (C#, C++, Dart, Go, Java, Python, Ruby, etc.)



Implementação do *client*

class flwr.client.NumPyClient

abstract evaluate(parameters: List, config)

Evaluate the provided weights using the locally held dataset.

PARAMETERS

- **parameters** – The current (global) model parameters
- **config** – Configuration parameters which allow the server to influence evaluation on the client. It can be used to communicate arbitrary values from the server to the client, for example, to influence the number of examples used for evaluation.

RETURNS

- **loss** – The evaluation loss of the model on the local dataset.
- **num_examples** – The number of examples used for evaluation.
- **metrics** – A dictionary mapping arbitrary string keys to values of type bool, bytes, float, int, or str. It can be used to communicate arbitrary values back to the server.

abstract fit(parameters: List, config)

Train the provided parameters using the locally held dataset.

PARAMETERS

- **parameters** – The current (global) model parameters.
- **config** – Configuration parameters which allow the server to influence training on the client. It can be used to communicate arbitrary values from the server to the client, for example, to set the number of (local) training epochs.

RETURNS

- **parameters** – The locally updated model parameters.
- **num_examples** – The number of examples used for training.
- **metrics** – A dictionary mapping arbitrary string keys to values of type bool, bytes, float, int, or str. It can be used to communicate arbitrary values back to the server.

abstract get_parameters()

RETURNS

- parameters – The local model parameters

Exemplo de descrição de interface

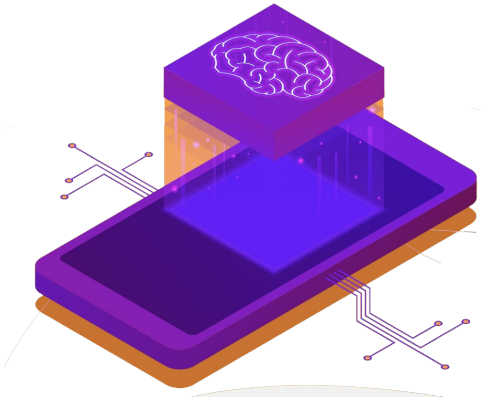
```
message ServerMessage {
    message Reconnect { int64 seconds = 1; }
    message GetParameters {}
    message FitIns {
        Parameters parameters = 1;
        map<string, Scalar> config = 2;
    }

    message EvaluateIns {
        Parameters parameters = 1;
        map<string, Scalar> config = 2;
    }

   message PropertiesIns { map<string, Scalar> config = 1; }
    oneof msg {
        Reconnect reconnect = 1;
        GetParameters get_parameters = 2;
        FitIns fit_ins = 3;
        EvaluateIns evaluate_ins = 4;
        PropertiesIns properties_ins = 5;
    }
}
```

```
message ClientMessage {
    message Disconnect { Reason reason = 1; }
    message ParametersRes { Parameters parameters = 1; }
    message FitRes {
        Parameters parameters = 1;
        int64 num_examples = 2;
        int64 num_examples_ceil = 3 [ deprecated = true ];
        float fit_duration = 4 [ deprecated = true ];
        map<string, Scalar> metrics = 5;
    }
    message EvaluateRes {
        int64 num_examples = 1;
        float loss = 2;
        float accuracy = 3 [ deprecated = true ];
        map<string, Scalar> metrics = 4;
    }
    message PropertiesRes { map<string, Scalar> properties = 1; }
    oneof msg {
        Disconnect disconnect = 1;
        ParametersRes parameters_res = 2;
        FitRes fit_res = 3;
        EvaluateRes evaluate_res = 4;
        PropertiesRes properties_res = 5;
    }
}
```

Por que Flower?

	TensorFlow Federated	PySyft	Flower	OpenFL
Framework	TensorFlow	PyTorch	Qualquer um	Qualquer um
Conceito	Integrado	Integrado	Estático	Estático
Suporte a GPU	Não	Sim	Sim	Sim
Carregamento Remoto de Dados	Não	Não	Sim	Sim
Agregação Segura	Não	Sim	Em desenvolvimento	Sim
Privacidade Diferencial	Sim	Sim	Em desenvolvimento	Em desenvolvimento
Documentação	Nas páginas oficiais existem tutoriais com diferentes aplicações, mas mesmo assim é difícil de ser utilizado em projetos próprios	Alguns tutoriais incompletos, algumas funções deprecadas	Tutoriais completos com aplicações exemplo	Tutoriais disponíveis tanto para aplicações com TensorFlow quanto para o PyTorch
Comunidade de desenvolvimento (# de estrelas no Github)	1,7k + 162k do Tensorflow	7,8k + 53,1k do PyTorch	640	1,6k

Tabela 1: Tabela comparativa entre os *frameworks* analisados, baseada na desenvolvida por Durken [2].



EXPERIMENTO

DESCRIÇÃO

O modelo a ser treinado de forma distribuída é um modelo de classificação de atividades humanas a partir de dados de sensores de smartphones, problema denominado de HAR (human activity recognition).

O dataset é o MotionSense (*), que contém sinais do giroscópio e acelerômetro de smartphones.

- Atividades: subindo e descendo escada, andando, correndo, sentado e parado.

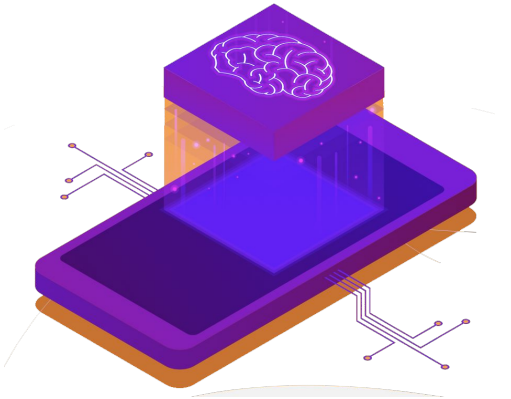
Serão configurados 4 clientes.

Tanto os clientes quanto o servidor de Federated Learning são executados na mesma máquina.

* https://www.kaggle.com/malekzadeh/motionsense-dataset

SIMPLIFICAÇÕES

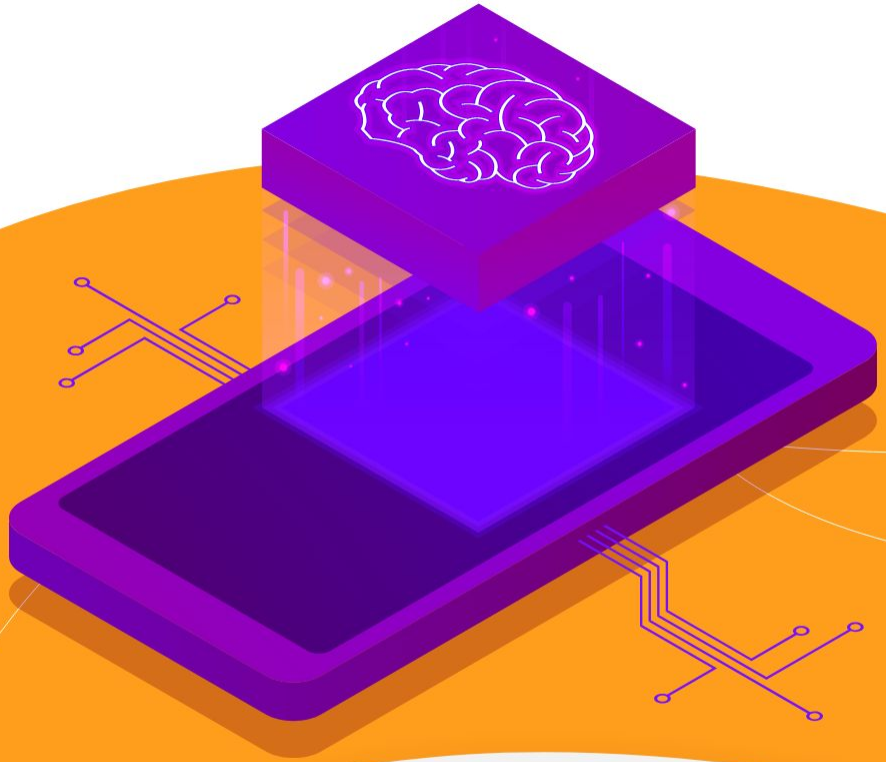
Fatores relacionados ao sistema, tais como a heterogeneidade na pilha de software, capacidades computacionais e largura de banda de comunicação, que afetam o treino e a sincronização dos modelos, são desconsiderados.



REFERÊNCIAS

REFERÊNCIAS

- [1] ABDULRAHMAN, Sawsan et al. A survey on federated learning: The journey from centralized to distributed on-site learning and beyond. IEEE Internet of Things Journal, v. 8, n. 7, p. 5476-5497, 2020.
- [2] MCMAHAN, Brendan et al. Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. PMLR, 2017. p. 1273-1282.
- [3]JERONYMO, Victor. Implementando métodos de otimizaçao para treinamento de redes neurais com Pytorch. 2019. Tese de Doutorado. Universidade Estadual de Campinas.



H.IAAC | HUB DE INTELIGÊNCIA ARTIFICIAL E ARQUITETURAS COGNITIVAS

Obrigado!

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÕES
PÁTRIA AMADA BRASIL GOVERNO FEDERAL
Softex
ELDORADO INSPIRAÇÃO PELO NOVO
UNICAMP
Instituto de Computação UNIVERSIDADE ESTADUAL DE CAMPINAS
FEEC