UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

FERNANDA CAVALHEIRO ORLANDI

Mitigando o impacto de dados *non*-IID em Federated Learning com Entropia

Dissertação de Mestrado

Orientador: Prof. Dr. Cláudio Fernando Resin Geyer

Co-orientador: Prof. Dr. Julio César Santos dos Anjos

Porto Alegre
2023

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

Orlandi, Fernanda Cavalheiro

Mitigando o impacto de dados *non*-IID em Federated Learning com Entropia / Fernanda Cavalheiro Orlandi. – Porto Alegre: PPGC da UFRGS, 2023.

93 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2023. Orientador: Claudio Fernando Resin Geyer. Co-orientador: Julio César Santos dos Anjos.

1. Federated Learning. 2. Collaborative Learning. 3. Machine Learning. 4. Big Data. 5. Heterogeneous Data. 6. Non-IID. I. Geyer, Claudio Fernando Resin. II. Anjos, Julio César Santos dos. III. Título.

"*E não nos cansemos de fazer o bem, pois no tempo próprio colheremos, se não desanimarmos.*"

— Gálatas 6:9

AGRADECIMENTOS

Agradeço a Deus por ter me dado inspiração e ânimo para persistir nesse programa de pós-graduação em computação.

Agradeço à minha família, pelo amparo à todas as minhas necessidades materiais e emocionais, principalmente minha Vó Jandira Ferreira da Silva, que sempre me deu apoio.

Agradeço aos colegas do Grupo de Processamento Paralelo e Distribuído (GPPD): Prof. Dr. Claudio R. Geyer por sua orientação e pela paciência que teve comigo, bem como pelas dicas e contatos que disponibilizou com outros pesquisadores. Dr. Julio César S. dos Anjos, por sua coorientação, paciência e votos de confiança nos momentos de turbulência.

Extendo meus agradecimentos à Universidade Federal do Rio Grande do Sul (UFRGS), pela hegemônica conduta em pesquisas científicas, que é exemplo primordial da importância da educação no Rio Grande do Sul, no Brasil e no mundo.

Agradeço à minha grande amiga Aline Vargas Nunes pelo companherismo, por ter passado as dificuldades comigo, dando o devido amparo para chegar ao final dessa jornada do curso de Mestrado.

RESUMO

Algoritmos de *Machine Learning* (ML) possibilitam processar um conjunto de dados de entradas para gerar coeficientes que ajustem a saída a um resultado previamente conhecido, como menor erro possível, fazendo com que seja possível reconhecer e extrair padrões de um grande volume de dados (*Big Data*). Isso permite construir um modelo de aprendizagem para tomada de decisão. Essa aprendizagem pode ser de forma colaborativa, onde a aprendizagem envolve grupos de indivíduos trabalhando juntos para resolver determinado problema. Essa abordagem chama-se *Collaborative Learning* e demonstra desempenho bastante otimizado em relação aos métodos tradicionais de ML em várias aplicações, como por exemplo, compreensão de imagem e reconhecimento de voz. Também é possível ter uma aprendizagem de máquina em ambiente federado, mais conhecido como *Federated Learning*, onde os dispositivos compartilham dados não sensíveis entre si, como seus parâmetros, ajustando o modelo no dispositivo e o modelo global, através de hiperparâmetros. No entanto, um modelo de *Federated Learning* pode sofrer com dados *non*-IID (não independentes e identicamente distribuídos), que podem ser dados heterogêneos, surgindo de diversas fontes de dados e dispositivos. Os dados *non*-IID causam baixa convergência para algoritmos de ML e alto consumo de energia, aumentando também a largura de banda. Um dos conceitos da Teoria da Informação, que é a entropia, serve para medir o grau de aleatoriedade dos dados. Este trabalho propõe um modelo de *Federated Learning* que mitiga o impacto dos dados *non*-IID por meio de um algoritmo FedAvg-BE, que fornece aprendizado federado com a avaliação de entropia de borda para selecionar dados com melhor qualidade, em um ambiente de dados *non*-IID. A avaliação do desempenho do algoritmo, no melhor caso, demonstra 26% de economia de tempo de execução do modelo proposto em configurações de FL para *datasets* conhecidos da literatura. Os resultados dos 115 experimentos realizados neste trabalho demonstram a viabilidade do modelo proposto para mitigar o impacto dos dados *non*-IID.

Palavras-chave: Big Data, Collaborative Learning, Federated Learning, Machine Learning, Dados heterogêneos, *Non*-IID.

Mitigating non-IID Data impact in Federated Learning with Entropy

ABSTRACT

Machine Learning (ML) algorithms make it possible to process a set of input data to generate coefficients that adjust the output to a previously known result, with the smallest possible error, making it possible to recognize and extract patterns from a large volume of data (Big Data). This will allow building a learning model for decision making. This learning can be collaborative, where learning involves groups of individuals working together to solve a given problem. This approach is called Collaborative Learning and demonstrates highly optimized performance compared to traditional ML methods in several applications, such as image understanding and voice recognition. It is also possible to have machine learning in a federated environment, better known as Federated Learning, where IoT devices share non-sensitive data with each other, such as their parameters, adjusting the model on the device and the global model, through hyperparameters. However, a Federated Learning model can suffer from non-IID (non-independent and identically distributed) data, which can be heterogeneous data, being produced from diverse data sources and devices. Non-IID data causes low convergence for ML algorithms and high power consumption, also increasing bandwidth. One of the concepts of Information Theory, which is entropy, serves to measure the degree of randomness of data. This work proposes a Federated Learning model that mitigates the impact of non-IID data through a FedAvg-BE algorithm, which provides federated learning with border entropy evaluation to select data with better quality, in a non-IID data environment. The evaluation of the performance of the algorithm, in the best case, demonstrates 26% of execution time savings of the proposed model in FL configurations for datasets known in the literature. The results of the 115 experiments carried out in this work demonstrate the viability of the proposed model to mitigate the impact of non-IID data.

Keywords: Big Data, Collaborative Learning, Federated Learning, Machine Learning, Heterogeneous Data, Non-IID.

LISTA DE ABREVIATURAS E SIGLAS

A-IoT	Autonomous Internet of Things
ACC	Accuracy
ADAM	Adaptive Moments
AI	Artificial Intelligence
APP	Application
ARM	Advanced RISC Machine
BatchNorm	Batch Normalization
CEI	Centro de Empreendimentos em Informática
CIFAR	Canadian Institute For Advanced Research
CL	Collaborative Learning
CNN	Convolutional Neural Networks
Conv	Convolutional
CPU	Central Processing Unit
DL	Deep Learning
FL	Federated Learning
FedAvg	Federated Averaging
FedAvg-BE	Federated Averaging with Border Entropy evaluation
GB	Gigabyte
GPU	Graphics Processing Unit
IDX	Index
IID	Independent and Identically Distributed
IoT	Internet of Things
LSTM	Long Short Term Memory
LTS	Long Term Support

MatLab	Matrix Laboratory
ML	Machine Learning
MLP	Multilayer Perceptron
MNIST	Modified National Institute of Standards and Technology
NLP	Natural Language Processing
NVFlare	NVIDIA Federated Learning Application Runtime Environment
RAM	Random Access Memory
ReLU	Rectified Linear Activation Function
RMSprop	Root Mean Squared Propagation
RNA	Rede Neural Artificial
RNN	Recurrent Neural Networks
RSL	Revisão Sistemática de Literatura
RvNN	Recursive neural networks
SCAFFOLD	Stochastic Controlled Averaging for Federated Learning
SDK	Software Development Kit
SGD	Stochastic Gradient Descent
Q&A	Question and Answer
TB	Terabyte
VCPU	Virtual CPU
VM	Virtual Machine

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A Aprendizagem Colaborativa, ou *Collaborative Learning*, é inspirada em uma abordagem educacional de ensino e aprendizagem que envolve grupos de alunos trabalhando juntos para resolver um problema, concluir uma tarefa ou criar um produto. Este conceito é expandido para a área de *Machine Leaning*, com uso de agentes que coletam informações do ambiente para aprenderem padrões segundo algum modelo estabelecido em uma arquitetura paralela e distribuída (SHAO *et al.*, 2019).

Machine Learning (ML) é uma das áreas mais importantes de Inteligência Artificial. Possibilita aprender um modelo ou um padrão de comportamento para determinada máquina executar tarefas. Um algoritmo de ML permite processar um conjunto de dados de entradas com os devidos padrões para gerar dados de saída. Assim, é possível reconhecer e extrair padrões de um grande volume de dados (*Big Data*), para construir um modelo de aprendizagem (CERRI *et al.*, 2017), servindo para a modelagem preditiva e tomadas de decisão.

Como é amplamente abordado na literatura quando se fala em *Big Data*, a produção de dados cresce de forma exponencial e rápida diariamente devido a várias fontes e novas tecnologias emergentes. Existem dispositivos inteligentes como componente principal, gerando grande volume de dados. Em *Machine Learning* e análises de dados, supõe-se que os dados são homogêneos, onde os eventos são independentes e identicamente distribuídos (IID), ou seja, com mesma distribuição de probabilidade e sem qualquer dependência entre eles. Isto é uma das premissas probabilísticas para o teste de hipóteses de eventos aleatórios. Contudo, objetos, valores, atributos e outros aspectos do dia a dia são essencialmente heterogêneos, isto é, são dados que têm uma distribuição de probabilidade onde os eventos não são independentes e identicamente distribuídos (*non*-IID) (CHIU *et al*, 2020). Os dados *non*-IID têm um alto grau de heterogeneidade e o conceito de entropia é uma medida que serve para mensurar a desordem nesses dados (PAVIOTTI, 2019).

A hipótese inicial é de que o uso do cálculo de entropia, selecionando dados de melhor qualidade com menor entropia, em modelos de *Collaborative Learning*, permita obter maior redução do tempo de execução do modelo, devido à troca de informações entre os dispositivos, em ambiente federado. Nesse tipo de ambiente, o aprendizado dos modelos também é chamado de *Federated Learning* (McMAHAN B. *et al.*, 2017).

O objetivo deste trabalho é verificar a influência da entropia na qualidade dos dados em ambiente IoT para reduzir o impacto de dados *non*-IID em algoritmos *Federated Learning*. Para isto, é necessário executar experimentos com esses algoritmos, usando conceitos e ferramentas de *Big Data* sobre um ambiente federado de sistema distribuído. Na literatura,

alguns artigos científicos foram revisados, pois apresentam estudos realizados sobre modelos de *Federated Learning*, a fim de identificar tipos de algoritmos e experimentos realizados para obter melhor percentual de acurácia sobre dados heterogêneos.

1.1 Definição do Problema

Em *Federated Learning* (McMAHAN *et al*, 2017), existe um problema em que os dados *non*-IID produzem um viés, também chamado *bias*, no cálculo dos parâmetros do algoritmo de otimização Stochastic Gradient Descent (SGD), durante o treinamento, causando a degradação da acurácia e aumentando tempo de execução do modelo, nos dispositivos da borda. Na medida em que o modelo local é ajustado em diferentes dispositivos com dados heterogêneos, a divergência entre os pesos do modelo global será acumulada, degradando o desempenho da aprendizagem.

A Figura 1 representa um gráfico de Acurácia vs. Número de Rodadas, como exemplo de acurácia obtida com distorções no treinamento, conforme o avanço do número de rodadas, usando dados *non*-IID, em comparação à acurácia com dados IID. No gráfico, a linha azul representa dados IID e a linha laranja representa dados *non*-IID. A linha laranja apresenta mais oscilações do que a linha azul, mostrando as distorções que dados *non*-IID podem causar, depreciando o resultado final da acurácia, ao finalizar as rodadas.

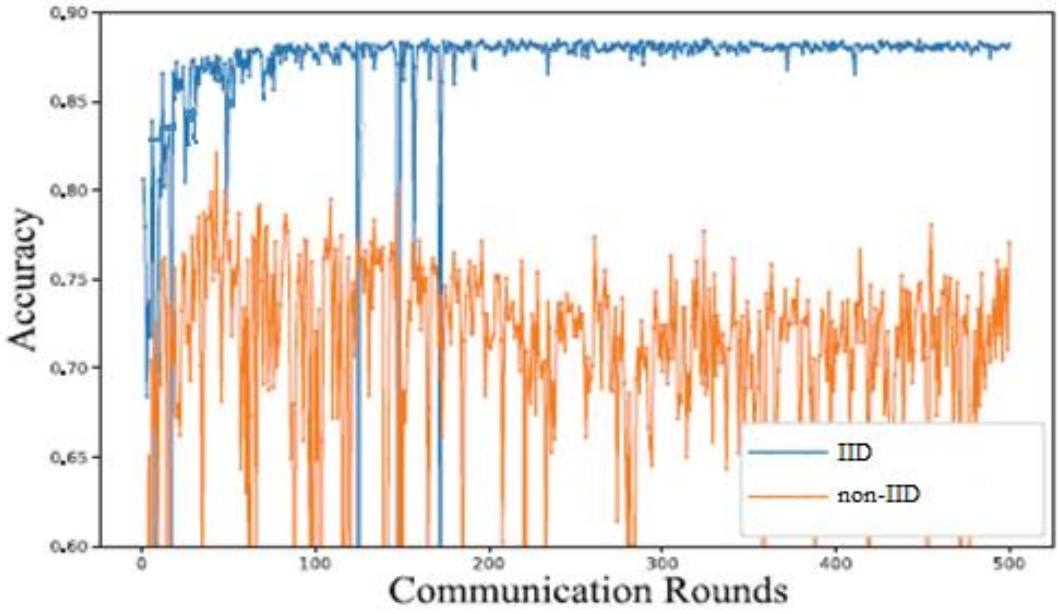


Figura 1: Exemplo de distorções na acurácia comparando dados IID vs. *non*-IID (adaptado de CHIU *et al*, 2020)

Além do exemplo citado, na Figura 1, seguem algumas situações negativas, que os dados *non*-IID ou heterogêneos podem causar (ZHU *et al.*, 2021):

- **Distorções** no treinamento do modelo;
- **Dificuldade** para convergência e desempenho adequado em Federated Learning;
- **Degradação** da acurácia e convergência mais lenta em algoritmos de otimização, por exemplo, o algoritmo SGD;
- **Divergência** dos hiperparâmetros durante o treinamento, diminuindo o desempenho do modelo;
- **Inconsistência** entre o algoritmo de otimização do dispositivo local e o modelo global, aumentando a perda geral da informação.

Com os recentes avanços tecnológicos e popularização dos dispositivos IoT, esse tipo de dispositivo tem possibilitado a coleta de dados heterogêneos, que requerem muita energia para processá-los. Portanto, desenvolver modelos para reduzir tempo de execução e consequentemente o consumo de energia, tratando os dados heterogêneos, mostra a devida importância para se mitigar a influência dos dados *non*-IID para ambientes de *Federated Learning*.

1.2 Motivação

Dispositivos inteligentes na borda, ou seja, que tem capacidade computacional suficiente para executarem algoritmos com inteligência artificial, trocam informações entre si e geram uma quantidade de dados em grande escala de aplicações de *Big Data*. Desses dados é possível reconhecer e extrair padrões para construir um modelo de aprendizagem. Nesse cenário espera-se obter dados homogêneos, porém os objetos, valores, atributos e outros aspectos do dia a dia são essencialmente heterogêneos, não independentes e distribuídos de forma idêntica (*non*-IID).

Nos últimos anos, o paradigma de computação de *Deep Learning* (DL) tornou-se gradativamente a abordagem computacional mais amplamente utilizada no campo de aprendizado de máquina, alcançando excelentes resultados em várias tarefas cognitivas complexas, igualando ou até superando aqueles fornecidos pelo desempenho humano. Uma das características do DL é a capacidade de aprender com grandes quantidades de dados (ALZUBAIDI *et al.*, 2021).

Em DL, o aprendizado centralizado tradicional exige que todos os dados coletados em dispositivos locais, como telefones celulares, sejam armazenados centralmente em um data center ou servidor em nuvem. Esse requisito aumenta o risco de privacidade e vazamento de dados, também impõe altas demandas de armazenamento e capacidade de computação do servidor quando a quantidade de dados é enorme (ZHU *et al.*, 2021).

O aprendizado de máquina distribuído é uma área de pesquisa relacionada a projetos de sistemas distribuídos. O procedimento de treinamento para um modelo de aprendizado de máquina distribuído é o seguinte: cada dispositivo realiza cálculos sobre os parâmetros e dados do modelo local, produzindo algumas estatísticas. Essas estatísticas são agregadas por meio de algum canal de comunicação e assim os parâmetros do modelo local são atualizados usando estatísticas agregadas (JIANG *et al.*, 2022). O paralelismo de dados distribuídos permite que várias máquinas treinem uma réplica de modelo com diferentes grupos de dados em paralelo e serve como uma solução potencial para o problema de armazenamento e capacidade computacional, porém ele ainda precisa de acesso a todos os dados de treinamento para dividi-los em fragmentos distribuídos de maneira uniforme, causando possíveis problemas de segurança e privacidade aos dados (ZHU *et al.*, 2021).

Federated Learning visa treinar um modelo global que pode ser treinado em dados distribuídos em diferentes dispositivos, protegendo a privacidade dos dados. Em 2016, pesquisadores, pela primeira vez, introduziram o conceito de aprendizado federado baseado no paralelismo de dados e propuseram um algoritmo de média federada, chamado FedAvg (McMAHAN *et al*, 2017). Assim, é essencial avaliar o desempenho de modelos de *Federated Learning*, com dados *non*-IID, sob uma configuração experimental metodológica adequada.

A motivação deste trabalho é realizar uma avaliação para estudar a hipótese de que existe uma relação entre entropia e dados *non*-IID, em um ambiente de *Federated Learning*, para comparar acurácia e tempo de execução. Nessa avaliação, pretende-se usar um ambiente que simula dispositivos de borda. Esse tipo de dispositivo lida com muitos dados e requer muita energia para processá-los, assim reduzir tempo de execução e consequentemente reduzir o consumo de energia em redes com dispositivos distribuídos em ambiente federado é um objetivo que vale a pena ser explorado.

1.3 Objetivos e Contribuições

O principal objetivo deste trabalho é estudar a hipótese de que o cálculo de entropia pode ter uma relação direta com dados *non*-IID, para otimizar o cálculo do algoritmo de otimização SGD e o desempenho, diminuindo o tempo de execução das tarefas em ambiente IoT sob algoritmos de Federated Learning. Para estudar esta hipótese, os objetivos específicos do trabalho são:

- Identificar a influência da entropia sobre dados *non*-IID em ambiente de *Federated Learning*;

- Elaborar um mecanismo consistente que seja independente do tipo de dataset para ajustar a entrada de dados ao processamento eficiente de dispositivos IoT;
- Elaborar uma metodologia de avaliação e seleção de dados que não cause um grande impacto na comunicação existente entre os dispositivos no ambiente federado;
- Desenvolver um modelo de solução que permita a criação de aplicações Federated Learning que avaliem a qualidade dos dados para ser processado somente o que agregar maior valor à informação;
- Criar um algoritmo para o tratamento da informação que estenda o modelo de aplicações Federated Learning mitigando os efeitos de dados *non*-IID.

Como contribuição adicional, este trabalho é uma pesquisa experimental que visa avaliar combinações de frameworks, modelos e conjuntos de dados *non*-IID em um ambiente de treinamento de *Federated Learning*. Este cenário de experimentação concentra-se na realização de cargas de treinamento em máquinas virtuais, com configuração semelhante a um dispositivo móvel, para comparar a acurácia e tempo de execução, verificando discrepâncias de treinamento e limitações de eficiência. Foi desenvolvido um modelo de *Federated Learning*, através do algoritmo *Federated Averaging with Border Entropy evaluation* (FedAvg-BE), que usa o cálculo de entropia para selecionar os dados, fazendo avaliação de entropia na borda (ORLANDI *et al*, 2023).

A metodologia deste trabalho também se destaca como uma contribuição, pois serve para avaliar a viabilidade de treinar dados *non*-IID, considerando entropia para selecionar os dados, em um modelo de *Federated Learning*. Além disso, este trabalho avalia e compara a acurácia e tempo de execução desse modelo com outros modelos da literatura em implementações de aplicações, usando o kit de desenvolvimento de software (SDK) NVFlare da plataforma NVIDIA. Os experimentos deste trabalho mediram o impacto de dados *non*-IID com cálculo de entropia no treinamento, validando modelos de *Federated Learning* quando processados com o *framework* PyTorch e SDK NVFlare, em relação a algumas métricas de software, como tempo total de execução, acurácia e função de perda.

Por fim, este trabalho também contribuiu elaborando uma revisão sistemática de literatura, pesquisando modelos existentes de *Federated Learning*, como também definições de configuração da arquitetura do mecanismo de rede para auxiliar os futuros usuários do SDK NVFlare na instalação e configuração dos ambientes em máquinas virtuais, auxiliando quem deseja obter os códigos-fonte para pesquisas experimentais com modelos FL.

1.4 Organização do Texto

Após esta introdução, a estrutura desta dissertação de mestrado está organizada como segue. O capítulo 2 apresenta os fundamentos teóricos, para estudos e execução dos experimentos, com conceitos e definições sobre *Big Data*, *Machine Learning e Federated Learning*, bem como os detalhes sobre teoria da informação, cálculo de entropia, tipos de Redes Neurais e algoritmos de otimização. O capítulo 3 aborda uma visão geral de modelos de *Federated Learning* e trabalhos relacionados, bem como o desenvolvimento de uma revisão sistemática de literatura, apresentando a metodologia de busca de artigos e um detalhamento sobre os artigos selecionados que englobam o estado da arte de modelos de Federated Learning. O capítulo 4 descreve o modelo FL proposto e suas etapas, como também o algoritmo FedAvg-BE, contribuindo para o estudo desta dissertação. No capítulo 5 é apresentado o plano de trabalho e metodologia, detalhando a implementação das aplicações para os 4 modelos selecionados e o modelo proposto com algoritmo de entropia na borda, FedAvg-BE. O capítulo 6 apresenta os 115 experimentos realizados e avaliação de resultados obtidos dos mesmos, mostrando comparativo de acurácia, tempo de execução e função de perda, entre as aplicações dos modelos. E, por fim, no capítulo 7 são apresentadas as considerações finais da pesquisa realizada, bem como os trabalhos futuros.

2 REFERENCIAL TEÓRICO

O objetivo deste capítulo é apresentar os conceitos essenciais das três áreas principais estudadas neste trabalho: *Big Data*, *Machine Learning* e *Federated Learning*. O conceito de *Big Data* está relacionado aos dados com maior variedade que atingem volumes e velocidade crescentes. *Machine Learning* é uma área de análise de dados e informações de forma que os sistemas aprendam por conta própria, reduzindo a necessidade de intervenção humana. *Federated Learning* permite que os dispositivos aprendam de forma colaborativa um modelo de aprendizado de máquina compartilhado, mantendo a privacidade dos dados. Nas seções a seguir, serão apresentados os principais conceitos de cada área.

2.1 *Big Data*

Muitos dispositivos inteligentes geram um volume enorme de dados. Esses dispositivos inteligentes são o componente principal de cidades inteligentes (inclui saúde inteligente, casa inteligente, irrigação inteligente, escolas inteligentes e rede inteligente), agricultura inteligente, controle de poluição inteligente, carro inteligente e transporte inteligente. Os dados são gerados não apenas por esses dispositivos, mas também por dispositivos de ciências, negócios, economia e governo.

Os conceitos descritos abaixo sobre Big Data são usados para impulsionar o desempenho de aplicações usando uma grande quantidade de conjunto de dados. A produção de dados vem crescendo rapidamente dia a dia devido a várias fontes, pois novas tecnologias emergentes atuam como catalisadores no crescimento desses dados, onde atinge ritmo exponencial (PATGIRI, 2019). Os dados que são produzidos podem conter certo grau de aleatoriedade, ou seja, podem ser fracamente estruturados, devido ao tipo de coleta e uso de determinados dispositivos pessoais.

As características de Big Data para grande quantidade de informações são recursos de dados de alto volume, velocidade e variedade que requerem tipos práticos e criativos de manipulação de dados para um entendimento mais amplo das informações (MACHADO, 2018). Na literatura foi definido inicialmente como modelo 3Vs e posteriormente foi expandido para 5Vs com os seguintes conceitos:

- **Volume**: grande quantidade de dados, devido a várias fontes de dados.
- **Variedade**: fontes de dados variadas, aumentando a complexidade de análise.
- **Velocidade**: com enorme volume e variedade de dados, o processamento dos mesmos deve ser ágil para gerar informações úteis.

- **Veracidade**: garantia de que o dados gerados são verdadeiros e autênticos.
- **Valor**: o quanto as informações obtidas são úteis.

2.2 Dados IID e *non*-IID

Em estatística, um conjunto de variáveis aleatórias é independente e identicamente distribuído (i.i.d. ou IID), quando os dados têm a mesma distribuição de probabilidade e os mesmos são independentes uns dos outros, ou seja, os dados são homogêneos (CLAUSET, 2011).

Por exemplo, ao lançar uma moeda 100 vezes pode-se obter cara 53 vezes e coroa 47 vezes. Se lançar a mesma moeda novamente pela 101ª vez, pode-se obter uma cara ou uma coroa com uma probabilidade de 0,5 para cada. Isso significa que o resultado ainda será de cara ou coroa, mesmo se não salvar as informações sobre meus resultados anteriores.

Ter dados independentes e distribuídos de forma idêntica é uma das suposições comuns para aprendizado de máquina, procedimentos estatísticos e teste de hipóteses. Essa suposição pode ser útil em tarefas de análise de dados, mesmo quando os dados não são rigorosamente IID.

Os dados *non*-IID são a suposição de que os dados são desestruturados ou fracamente estruturados, com alto grau de heterogeneidade. Objetos, valores, atributos e outros aspectos do dia a dia são essencialmente não independentes e identicamente distribuídos (*non*-IID) (CHIU *et al*, 2020).

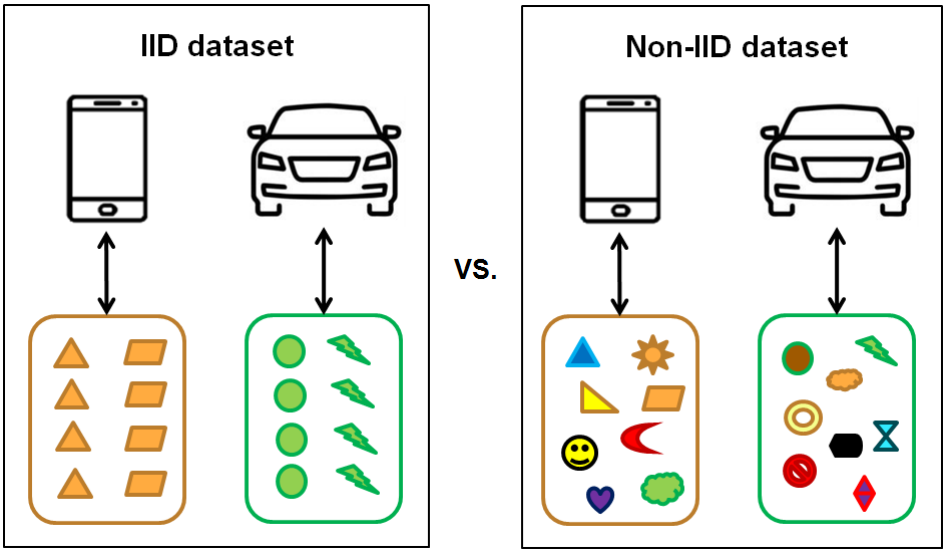


Figura 2: Ilustração de dados IID vs. *non*-IID

2.3 Teoria da Informação e Entropia

A Teoria da Informação é um ramo da matemática aplicada que aborda sobre a quantificação e comunicação da informação. Foi criada para estudar o envio de mensagens de textos em um canal ruidoso, como uma comunicação por transmissão de rádio. No contexto de matemática, essa teoria explica como projetar códigos ótimos e calcular o comprimento esperado de mensagens amostradas a partir de distribuições de probabilidade específicas, usando vários esquemas de codificação (GOODFELLOW *et al*, 2017).

No contexto da computação, também podemos aplicar a teoria da informação a variáveis contínuas, onde a interpretação de comprimento de mensagem não se aplica, mas são usadas algumas ideias da teoria da informação para caracterizar as distribuições de probabilidade ou quantificar a similaridade entre as distribuições de probabilidade.

O conceito de entropia é o elo que liga a Teoria da Informação com outras áreas de ciência. A fórmula da entropia foi explicada no artigo "Uma teoria matemática de comunicação" em 1948, de Claude E. Shannon. Em computação, a entropia é uma medida bem definida da desordem ou informação encontrada nos dados, ou seja, mede o grau de aleatoriedade de uma variável (PAVIOTTI, 2019). Pode-se dizer que quanto maior a entropia, mais os dados são *non*-IID. A Figura 3 mostra um exemplo visual para representar a entropia. Supondo que existem três potes, um com bolinhas vermelhas, outro com bolinhas amarelas e o terceiro vazio. No pote vazio enche-se com as bolinhas amarelas e por cima as bolinhas vermelhas, separadas e organizadas por cor. Ao balançar o pote, as bolinhas se misturam de uma forma que não existe mais separação e dificilmente voltarão ao estado inicial organizadas. Ou seja, se tornou um sistema com desordem, significando aumento de entropia.

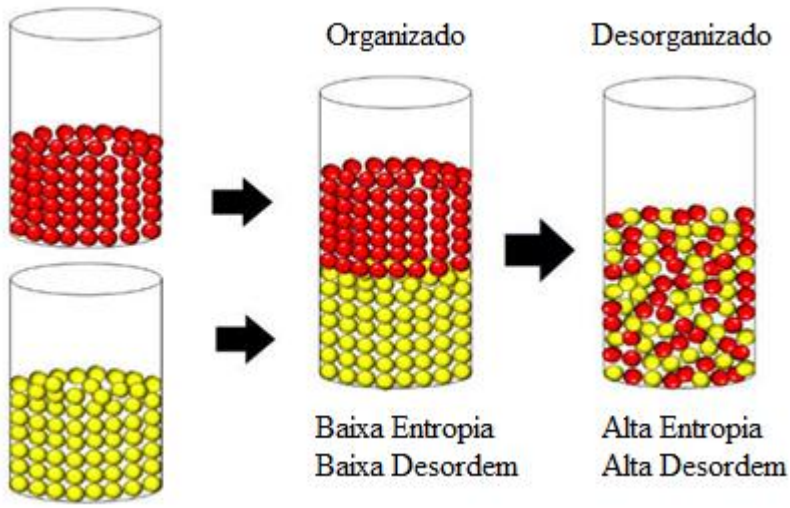


Figura 3: Exemplo de entropia com bolinhas em potes (ORLANDI *et al.*, 2023)

O gráfico da Figura 4 mostra entropia para uma base de dados em que a classificação é binária, com dois valores possíveis. Também é possível calcular entropia para qualquer quantidade de valores. O eixo X representa a probabilidade de uma das classes ocorrer no *dataset*. O eixo Y é o resultado da entropia do *dataset*. Na probabilidade de distribuição, no limite inferior, quando ***p*** está próximo de 0, a entropia *H*(***p***(0)) tende ser 0 e no limite superior quando ***p*** está próximo de 1, a entropia *H*(***p***(1)) também tender ser 0. Quando ***p*** = 0,5, a entropia *H*(***p***(0,5)) é máxima.

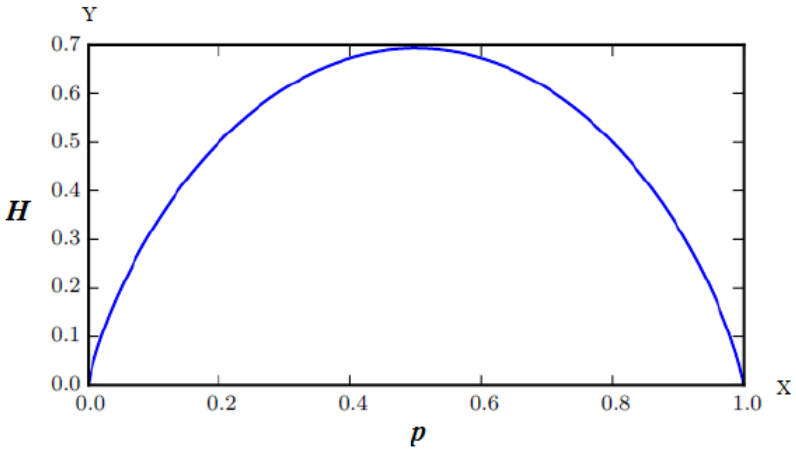


Figura 4: Gráfico de entropia para classificação binária (adaptado de GOODFELLOW *et al*, 2017)

Podemos quantificar a quantidade de incerteza em uma distribuição de probabilidade inteira usando a fórmula de entropia de Shannon. Assim sendo, a entropia de Shannon de uma distribuição é a quantidade esperada de informações em um evento extraída dessa distribuição, como ilutrada na fórmula abaixo.

$$H = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

Equação (1)

Onde:

H = Entropia, medida do grau de incerteza da informação.

p*i* = Probabilidade de distribuição, a qual uma fração de registros pode pertencer à classe ***i*** em um *dataset*.

2.4 *Machine Learning*

Machine Learning (ML) é uma das subáreas mais importantes de Inteligência Artificial. Ela possibilita encontrar nos dados um modelo ou um padrão de comportamento para determinada máquina. Ou seja, um algoritmo de Machine Learning possibilita processar

um conjunto de dados de entradas com os devidos padrões para depois inferir a saída, tendo como base um conjunto de dados não avaliados previamente. Assim, é possível reconhecer e extrair padrões de um grande volume de dados, o que permite construir um modelo de aprendizado (CERRI, 2017).

A Figura 5 mostra o processo de execução de um algoritmo de Machine Learning. O processo se inicia com entrada de dados, que são fundamentais, pois sem eles seria impossível criação de algoritmos dessa natureza. Os dados podem estar de formas variadas: completos, com erros ou faltantes. Dependendo do tipo de aprendizagem de máquina, esses dados devem estar classificados e catalogados de maneira correta, exigindo maior tempo de pré-processamento. Algoritmos de machine learning requerem grande quantidade de dados, se possível, completos, mas quanto maior for o conjunto de dados maior será o custo computacional.

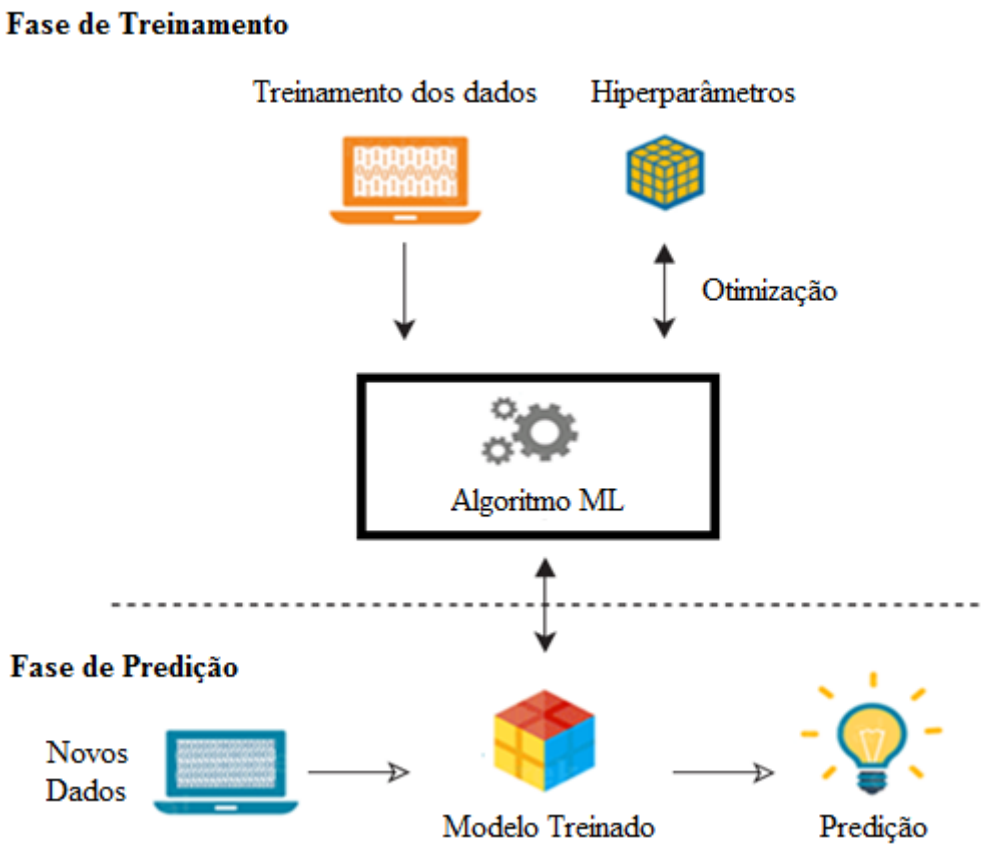


Figura 5: Processo de execução de um algoritmo de Machine Learning

É importante saber identificar e selecionar os atributos mais importantes, pois esses atributos serão responsáveis para gerar um modelo de conhecimento consistente. Nessa seleção de atributos, é comum excluir certos dados para não causar problema no treinamento do modelo, pois traz melhorias de tempo, performance e aprendizagem (SHEENA, 2016).

Cada conjunto de dados tem características próprias, que precisa de modelos de tipos específicos para ter boa acurácia. Assim como conjunto de dados, cada algoritmo funciona de maneira diferente, com custos computacionais e lógicos diferentes. Na figura 6, são apresentadas as abordagens de Machine Learning. De acordo com (VERBRAEKEN *et al*, 2021) e (GOODFELLOW *et al*, 2017), as abordagens de aprendizado de máquina são as seguintes:

- Aprendizado **Supervisionado** – Esse tipo de abordagem está associado a um rótulo ou destino. Usa dados de treinamento que consistem em objetos de entrada (geralmente vetores) e os valores de saída desejados correspondentes para aprender a classificar o rótulo correspondente. Por exemplo, o *dataset* Iris possui espécies de cada planta de íris. O algoritmo do tipo supervisionado aprende a classificar as plantas de íris em três espécies diferentes.
- Aprendizado **Não supervisionado** – Trabalha com dados não rotulados. Usa dados de treinamento que consistem em objetos de entrada (geralmente vetores) sem valores de saída. Esse tipo de algoritmo visa encontrar uma função que descreva a estrutura dos dados e agrupe os dados de entrada não classificados. O caso de uso mais comum é agrupar dados com base em semelhanças e padrões ocultos. Alguns algoritmos deste tipo de aprendizado desempenham outras funções, como *clustering*, que consiste em dividir o conjunto de dados em clusters de exemplos semelhantes.
- Aprendizado **Semissupervisionado** – Trabalha com uma quantidade (geralmente pequena) de dados rotulados, complementados por uma quantidade comparativamente grande de dados não rotulados. O *clustering* pode ser usado para extrapolar rótulos conhecidos em pontos de dados não rotulados.
- Aprendizado **Por Reforço** – Os algoritmos desse tipo de abordagem não experimentam apenas um conjunto de dados fixo. Esses algoritmos interagem com um ambiente, de modo que um agente é treinado e deve realizar ações com base em suas observações. As ações tomadas recebem *feedbacks*, que dependem de uma função de recompensa ou custo que avalia os estados do sistema.

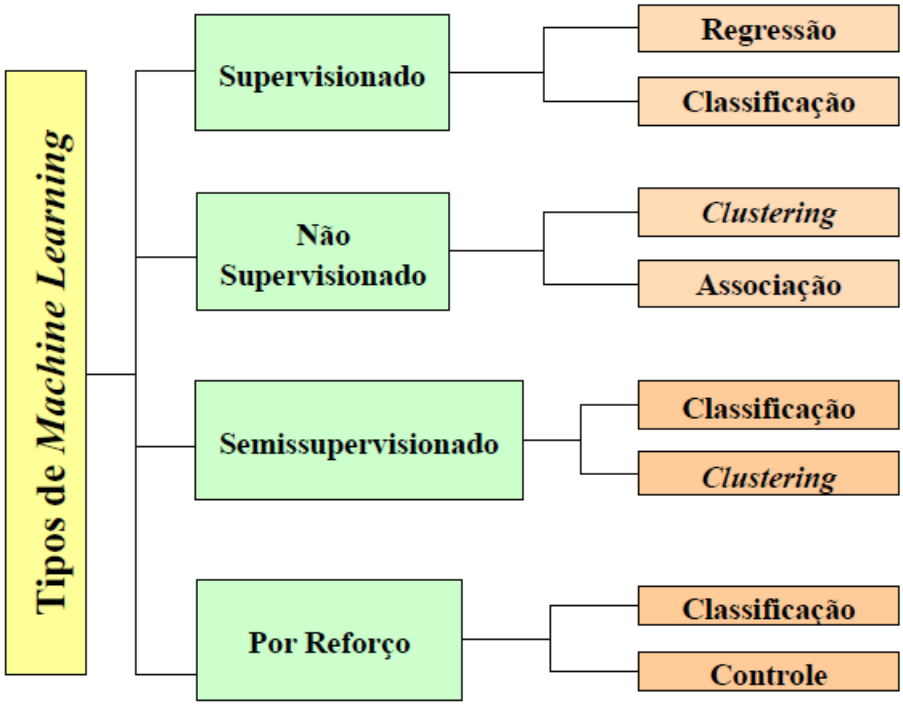


Figura 6: Tipos de *Machine Learning*

O treinamento dos dados é um recurso computacional capaz de criar um modelo baseado nesses mesmos dados, fazendo com que tenha capacidade de classificar ou predizer novos dados.

Em algoritmos de *Machine Learning*, é usado um método chamado de avaliação aplicado ao modelo para verificar sua capacidade e testar esse modelo em dados nunca processados antes, verificando a porcentagem ou a aproximação da predição do modelo. Existem diversas técnicas para fazer a avaliação, cada técnica é adequada ao tipo de dados (CATAL, 2012).

2.4.1 *Deep Learning*

Deep Learning (DL) é uma subárea de *Machine Learning*, como mostra a Figura 7. As técnicas dessa subárea permitem que os sistemas de computador melhorem com a experiência e o processamento dos dados (GOODFELLOW *et al*, 2017). É uma abordagem viável para a construção de sistemas de inteligência artificial que podem operar em ambientes complexos do mundo real. Além disso, usa uma grande quantidade de dados para mapear a entrada fornecida para rótulos específicos (ALZUBAIDI *et al*, 2021).

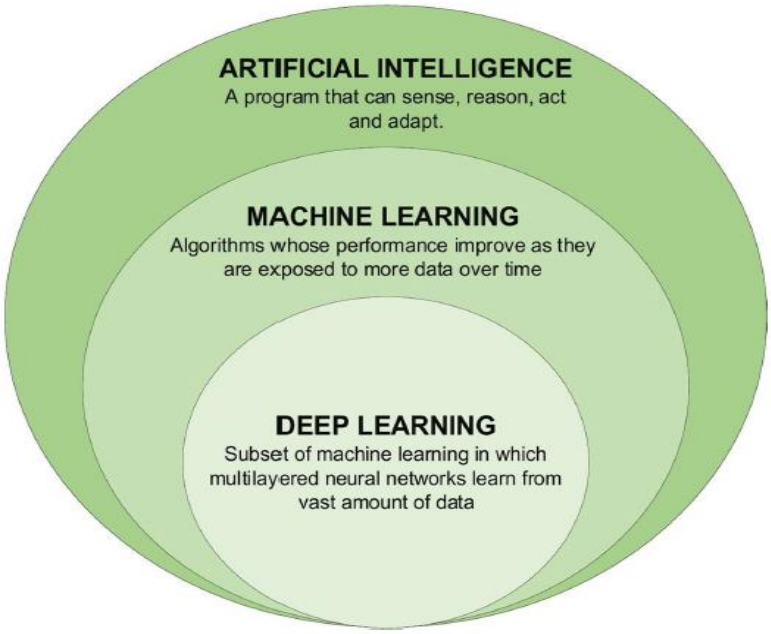


Figura 7: Subáreas de Inteligência Artificial (ALZUBAIDI *et al*, 2021)

Deep Learning alcança grande poder e flexibilidade ao aprender a representar o mundo como uma hierarquia aninhada de conceitos, com cada conceito definido em relação a conceitos mais simples e representações mais abstratas computadas em termos de conceitos menos abstratos (GOODFELLOW *et al*, 2017).

Alguns dos primeiros algoritmos de aprendizado que reconhecemos hoje pretendiam ser modelos computacionais de aprendizado biológico, ou seja, modelos de como o aprendizado acontece ou poderia acontecer no cérebro. Com isso, um dos nomes que DL possui é redes neurais artificiais (RNAs). A perspectiva neural sobre aprendizagem profunda é motivada por duas perspectivas principais. Uma perspectiva é que o cérebro fornece uma prova, por exemplo, de que o comportamento inteligente é possível. Um caminho conceitualmente simples para construir inteligência é fazer engenharia reversa dos princípios computacionais por trás do cérebro e duplicar sua funcionalidade. Outra perspectiva é que seria profundamente interessante entender o cérebro e os princípios que fundamentam a inteligência humana, de modo que os modelos de aprendizado de máquina que esclarecem essas questões científicas básicas são úteis além de sua capacidade de resolver aplicativos de engenharia.

A maioria das indústrias e negócios já sofreu disrupção e transformação através do uso de DL. As principais empresas focadas em tecnologia e economia em todo o mundo estão em uma corrida para melhorar o DL. Mesmo agora, o desempenho e a capacidade do nível humano não podem exceder o desempenho do DL em muitas áreas, como prever o tempo necessário para fazer entregas de carros, decisões para certificar solicitações de empréstimos e prever classificações de filmes. Ainda há mais avanços a serem feitos no contexto de educação a distância (EAD). DL fornece melhorias para a vida humana, sendo muito útil para cálculos

de precisão, que inclui estimativas de desastres naturais, descoberta de medicamentos e diagnóstico de câncer. Em 2020, DL tornou-se a principal ferramenta em muitos hospitais ao redor do mundo para classificação e detecção automática de COVID-19 usando imagens de radiografia de tórax ou outros tipos de imagens (ALZUBAIDI *et al*, 2021).

Redes Neurais Artificiais (RNAs) são sistemas baseados em modelos matemáticos de neurônios que consistem em várias camadas: uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída, como mostra a Figura 8. Cada camada consiste em neurônios conectados às camadas anterior e seguinte por meio de arestas com pesos associados (geralmente chamados de sinapses) e funções de ativação vinculadas ao neurônio (nós) que adicionam um comportamento não linear à sua saída como ReLU, Sigmoid, etc. (VERBRAEKEN *et al*, 2021).

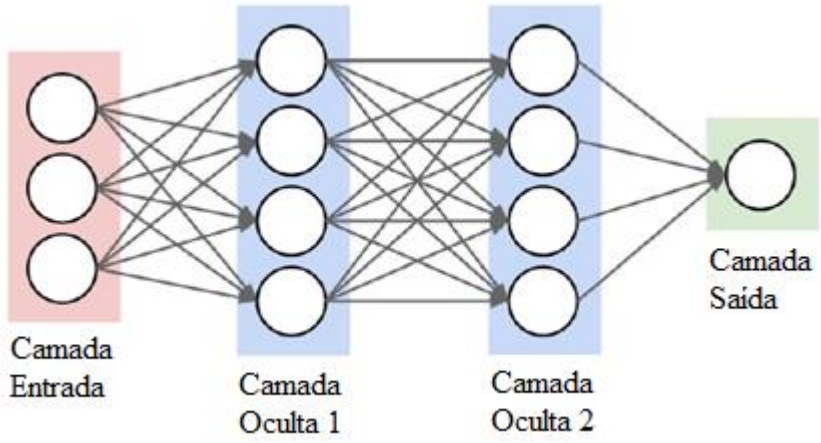


Figura 8: Representação de Rede Neural Artificial

As redes neurais são amplamente estudadas devido à sua capacidade de analisar enormes conjuntos de dados. Os tipos mais famosos de redes de aprendizado profundo incluem redes neurais recursivas (RvNNs), RNNs e CNNs, sendo CNN o mais utilizado em diversas aplicações dentre outras redes. De acordo com Alzubaidi *et al.* (2021), os tipos de rede serão explicados a seguir:

- Recursive neural networks (**RvNNs**) – A arquitetura RvNN é gerada para processar objetos, que possuem estruturas de formato aleatório como grafos ou árvores. Gera representação distribuída a partir de estrutura de dados recursiva de tamanho variável. A rede é treinada usando sistema de *back-propagation through structure* (BTS). Esse sistema funciona com a mesma técnica do algoritmo de Backpropagation e tem a capacidade de suportar uma estrutura no formato de árvore, como mostra a Figura 9.

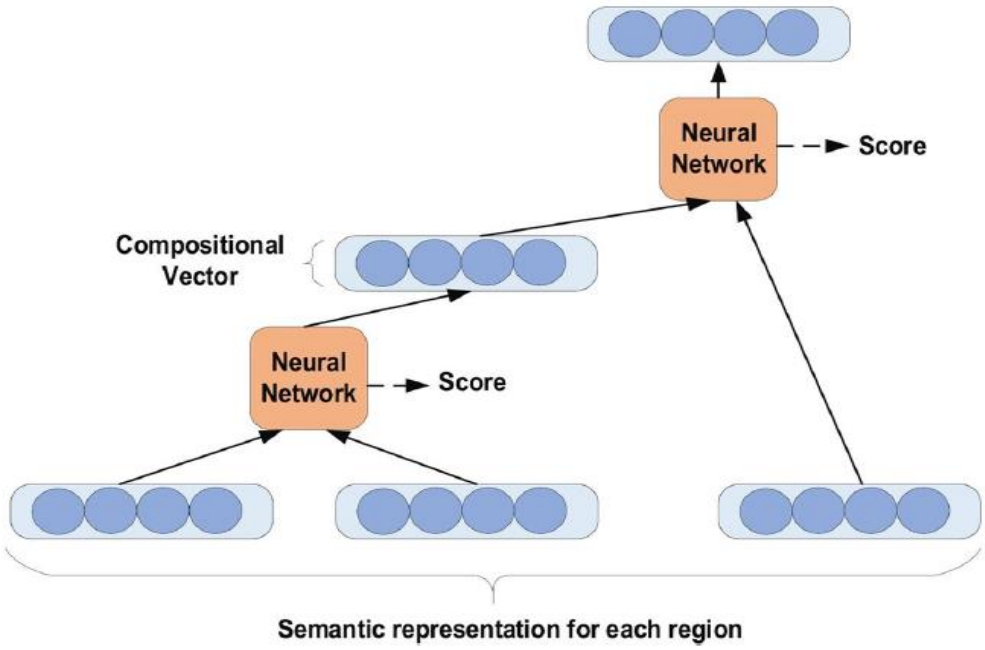


Figura 9: exemplo de árvore RvNN (ALZUBAIDI *et al*, 2021)

- Recurrent neural networks (**RNNs**) – RNN é um algoritmo muito utilizado em DL e usa dados sequenciais na rede. Aplica-se na área de processamento da fala e contextos de processamento da linguagem natural (NLP). Por exemplo, procura-se entender o contexto da frase para determinar o significado de uma palavra específica dentro desta mesma frase, como uma unidade de memória de curto prazo. As sinapses recorrentes dão à rede uma memória. Isso pode ajudar na descoberta de padrões temporais nos dados. Blocos de neurônios em redes recorrentes operam como células com memórias distintas e podem armazenar informações por um período de tempo arbitrariamente longo. Essa abordagem oferece conexões recorrentes a blocos de memória na rede. Cada bloco de memória contém um número de células, que têm a capacidade de armazenar os estados temporais da rede. A figura 10 ilustra um diagrama de algoritmo RNN.

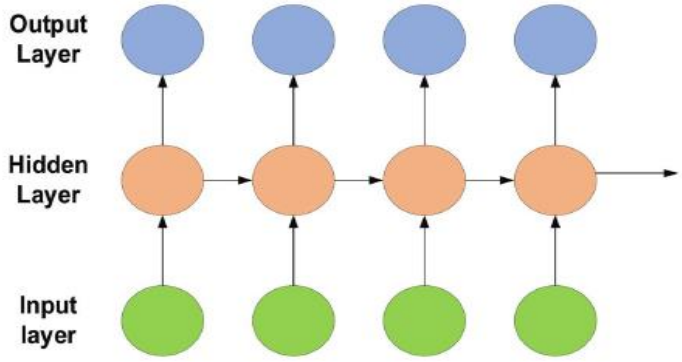


Figura 10: Diagrama de RNN (ALZUBAIDI *et al*, 2021)

- Convolutional neural networks (**CNNs**) – CNN é o tipo de algoritmo mais conhecido na literatura e muito usado em aplicações, pois identifica automaticamente os recursos relevantes sem qualquer supervisão humana. É um tipo de rede neural para processamento de dados que possui topologia do tipo grade. É tratado como uma forma de reconhecer características abstratas nos dados. A convolução faz com que a rede considere apenas dados locais. Esse algoritmo tem sido amplamente aplicado em diversos campos, incluindo visão computacional, processamento de fala, reconhecimento facial etc. As redes convolucionais são simplesmente redes neurais que usam a convolução no lugar da multiplicação geral de matrizes em pelo menos uma de suas camadas. O encadeamento de várias dessas camadas de convolução pode tornar a rede capaz de reconhecer estruturas complicadas em grandes conjuntos de dados. Exemplos disso são animais em imagens ou o significado contextual de uma frase em um parágrafo.

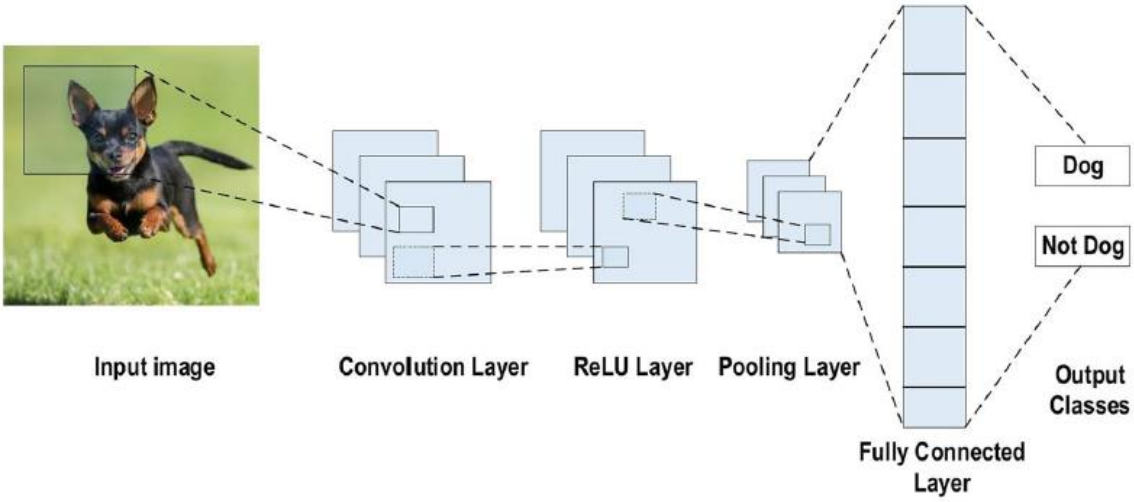


Figura 11: Exemplo de arquitetura CNN para classificação de imagem. (ALZUBAIDI *et al*, 2021)

Vários componentes são necessários para construir um modelo de Deep Learning, que devem ser combinados como um algoritmo ou técnica de otimização, uma função de custo, um modelo de rede neural e um conjunto de dados. Alguns algoritmos são usados para otimizar as redes neurais artificiais, como por exemplo SGD, Adam e RMSprop, que serão descritos nas subseções seguintes.

2.4.2 SGD

Stochastic Gradient Descent (SGD) é um algoritmo muito usado em vários algoritmos de aprendizado de máquina e forma a base para otimizar as redes neurais, sendo uma extensão

do algoritmo de *Gradient Descent* (GOODFELLOW *et al*, 2017). Basicamente, SGD é um conjunto de passos para atingir um mínimo da função de perda, como representado no gráfico da Figura 12. A função de perda ou *loss function* é uma função que mapeia um evento ou valores de uma ou mais variáveis num número real intuitivamente representando algum custo associado ao evento.

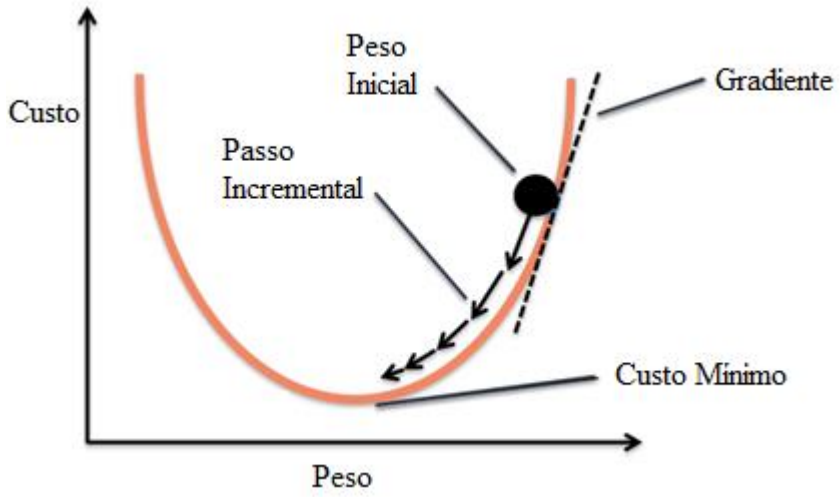


Figura 12: Ilustração do funcionamento do algoritmo Gradiente Descendente

Gradiente Descendente é um algoritmo iterativo que possui um número de passos com: parâmetros iniciais ou pesos θ *(theta)*, número de iterações *T*, taxa de aprendizagem *n* e local *L*, que computa hiperparâmetros da função de perda (mínima). Na iteração, computa-se o gradiente $L(\theta)$. O ponto central é a regra de iteração, que move o ponto do theta na direção do gradiente com uma taxa de aprendizagem. No algoritmo 1, é possível visualizar esses passos.

Algorithm 1 The general gradient descent algorithm.

Input: initial weights $\theta^{(0)}$, iterations T, learning rate η
Output: final weights $\theta^{(T)}$

1. **for** $t = 0$ **to** $T - 1$
2. compute $\nabla L(\theta^{(t)})$
3. $\theta^{(t+1)} := \theta^{(t)} - \eta \nabla L(\theta^{(t)})$
4. **return** $\theta^{(T)}$

Algoritmos baseados em SGD minimizam uma função de perda definida nas saídas do modelo adaptando os parâmetros do modelo na direção do gradiente negativo (isso representa a derivada multivariável de uma função). A descida do gradiente é chamada estocástica, pois o gradiente é calculado a partir de um subconjunto aleatório de amostras individuais dos dados de treinamento. O SGD não calcula o gradiente sobre todo o conjunto de dados numa única iteração (VERBRAEKEN *et al*, 2021). Uma amostra individual ***i*** é um bloco dos dados que é selecionado de forma aleatória em cada passo, conforme consta na Equação 2.

$$\theta^{(t+1)} = \theta^{(t)} - \eta \underbrace{\nabla L_i(\theta^{(t)})}_{\text{Individual samples}}$$

Equação (2)

Usar amostras individuais em cada passo faz com que SGD seja vantajoso, pois torna o algoritmo mais eficiente, introduzindo um ruído benéfico para o cálculo e ajuda a reduzir a redundância dos *datasets* reais. Porém, como desvantagem, as etapas para atingir uma mínima precisam de muitas atualizações, o que pode inclinar a descida em outras direções, onerando o tempo de execução para atingir a convergência até a mínima, tornando essas atualizações frequentes caras do ponto de vista computacional (BOTTOU *et al*, 2018). A seguir serão apresentados alguns dos mais significativos algoritmos de otimização no processo de computação do SGD.

2.4.3 Adam

Adam é um algoritmo de otimização adaptativo para taxa de aprendizado, pois o nome Adam deriva da frase "Adaptive Moments". O parâmetro momentum é incorporado diretamente como uma estimativa de primeira ordem, com peso exponencial, do gradiente, como apresentado no Algoritmo 2. Este algoritmo inclui correções de viés (bias) para estimativas de momentos de primeira e segunda ordem para contar na inicialização. É geralmente considerado robusto para escolha de hiperparâmetros, embora a taxa de aprendizado necessita ser alterada algumas vezes diferente do padrão sugerido (GOODFELLOW *et al*, 2017).

2.4.4 RMSprop

Root Mean Squared Propagation, ou RMSprop, é um algoritmo que tem um desempenho melhor na configuração não convexa, alterando o acúmulo de gradiente em uma média móvel ponderada exponencialmente. Alguns algoritmos quando aplicados a uma função não convexa para treinar a rede neural, o aprendizado pode passar por muitas estruturas diferentes e chegar em uma região onde é convexa, com isso a taxa de aprendizado pode ter se tornado pequena demais antes de chegar nessa região. O algoritmo RMSProp usa uma média exponencialmente decadente para descartar o histórico do passado extremo para que possa convergir rapidamente após encontrar uma região convexa, pois possui um hiperparâmetro que controla comprimento da média móvel (GOODFELLOW *et al*, 2017). O Algoritmo 3 apresenta seu funcionamento.

Algorithm 2 The Adam algorithm

Require: Step size ϵ (Suggested default: 0.001)
Require: Exponential decay rates for moment estimates, ρ_1 and ρ_2 in $[0, 1)$. (Suggested defaults: 0.9 and 0.999 respectively)
Require: Small constant δ used for numerical stabilization. (Suggested default: 10^{-8})
Require: Initial parameters $\boldsymbol{\theta}$
  Initialize 1st and 2nd moment variables $\boldsymbol{s} = \boldsymbol{0}$, $\boldsymbol{r} = \boldsymbol{0}$
  Initialize time step $t = 0$
  while stopping criterion not met **do**
    Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    $t \leftarrow t + 1$
    Update biased first moment estimate: $\boldsymbol{s} \leftarrow \rho_1 \boldsymbol{s} + (1 - \rho_1)\boldsymbol{g}$
    Update biased second moment estimate: $\boldsymbol{r} \leftarrow \rho_2 \boldsymbol{r} + (1 - \rho_2)\boldsymbol{g} \odot \boldsymbol{g}$
    Correct bias in first moment: $\hat{\boldsymbol{s}} \leftarrow \frac{\boldsymbol{s}}{1 - \rho_1^t}$
    Correct bias in second moment: $\hat{\boldsymbol{r}} \leftarrow \frac{\boldsymbol{r}}{1 - \rho_2^t}$
    Compute update: $\Delta\boldsymbol{\theta} = -\epsilon \frac{\hat{\boldsymbol{s}}}{\sqrt{\hat{\boldsymbol{r}}} + \delta}$ (operations applied element-wise)
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
  end while

Algorithm 3 The RMSProp algorithm

Require: Global learning rate ϵ, decay rate ρ.
Require: Initial parameter $\boldsymbol{\theta}$
Require: Small constant δ, usually 10^{-6}, used to stabilize division by small numbers.
  Initialize accumulation variables $\boldsymbol{r} = 0$
  while stopping criterion not met **do**
    Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    Accumulate squared gradient: $\boldsymbol{r} \leftarrow \rho \boldsymbol{r} + (1 - \rho)\boldsymbol{g} \odot \boldsymbol{g}$
    Compute parameter update: $\Delta\boldsymbol{\theta} = -\frac{\epsilon}{\sqrt{\delta + \boldsymbol{r}}} \odot \boldsymbol{g}$. ($\frac{1}{\sqrt{\delta + \boldsymbol{r}}}$ applied element-wise)
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
  end while

2.4.5 Técnicas de *Deep Learning*

Segundo Goodfellow (GOODFELLOW *et al*, 2017), muitas técnicas de otimização não são exatamente algoritmos, mas sim modelos gerais que podem ser especializados para gerar algoritmos. A técnica *Batch Normalization* é um método de reparametrização adaptativo, onde há dificuldade de treinar modelos de Deep Learning. A reparametrização reduz significativamente o problema de coordenar atualizações em várias camadas. Essa técnica pode ser aplicada a qualquer entrada ou camada oculta em uma rede neural. A técnica *Dropout* é uma estratégia de regularização, que fornece um método de baixo custo computacional para treinar e avaliar um conjunto de redes neurais de forma exponencial. O algoritmo de Dropout treina o conjunto de sub-redes, removendo unidades sem saída de uma rede original.

2.5 *Collaborative Learning*

Na era de *Big Data*, a informação tem baixo custo, mas o processamento dos dados custa caro. À medida que o poder computacional se aproxima das fontes de dados, as redes de sensores podem se tornar mais densas e poderosas. Uma rede de sensores é composta por um grande número de nós e consiste em componentes de detecção, processamento de dados e comunicação (DENNIS *et al.,* 2020). Os sensores nessas redes comunicam dados a um controlador logicamente centralizado que funde os dados para fornecer vários *insights* sobre as observações (NORDLUND *et al.*, 2019).

A abordagem de *Collaborative Learning* tem papel importante para desenvolver algoritmos para tomada de decisão preditiva com ferramentas *Big Data*, em um ambiente com sistemas distribuídos, que processa dados de sensores. Essa abordagem combina diferentes redes neurais em uma arquitetura de treinamento paralelo (WU *et al.*, 2020).

Alguns modelos de *Collaborative Learning* podem ser aplicados em outros tipos de ambientes, como por exemplo, ambiente de dispositivos autônomos de IoT (A-IoT). Dados podem ser processados e analisados por meio de técnicas de *Machine Learning*, com o objetivo de tomar decisões para controlar dispositivos de Internet das Coisas (IoT) ao mundo físico. IoT conecta um grande número de dispositivos à Internet, onde esses dispositivos geram uma grande quantidade de dados sensoriais para refletir o status do mundo físico (LEI *et al.*, 2020).

A abordagem A-IoT é uma combinação de análises e inteligência artificial, que torna os dispositivos autônomos. Os sistemas A-IoT um ambiente dinâmico e interativo, que detectam o ambiente e tomam decisões de controle. Nessa abordagem, o uso de inteligência artificial é fundamental, pois permite que os dados sejam processados localmente possibilitando uma

tomada de decisão mais rápida. Assim, consequentemente, pode-se diminuir o tráfego de rede e evitar-se possíveis sobrecargas de processamento em *Cloud* (LEI *et al.*, 2020).

2.5.1 *Distributed Machine Learning*

Pequenos modelos de aprendizado de máquina são treinados com quantidades modestas de dados, porém a quantidade de dados para treinar modelos maiores, como redes neurais, cresce exponencialmente com o número de parâmetros. Como a demanda por processamento de dados de treinamento ultrapassou o aumento do poder computacional de máquinas, há a necessidade de distribuir a carga de trabalho de aprendizado de máquina em várias máquinas e transformar o sistema centralizado em um sistema distribuído. Os sistemas distribuídos precisam ter um processamento de treinamento eficiente e criação de modelo coerente (VERBRAEKEN *et al*, 2021).

Os sistemas totalmente distribuídos (Figura 13) consistem em uma rede de nós independentes que agrupam a solução e onde nenhuma função específica é atribuída a determinados nós. Cada nó tem sua própria cópia dos parâmetros e os nós se comunicam diretamente entre si. A vantagem é ter uma escalabilidade tipicamente mais alta do que um modelo centralizado e a eliminação de pontos únicos de falha no sistema.

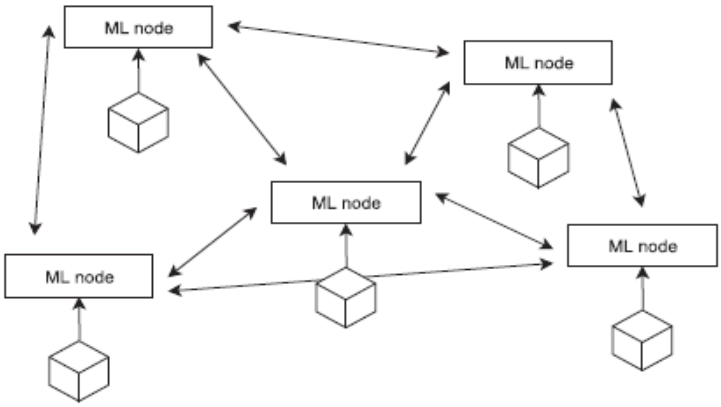


Figura 13: Sistema distribuído de *Machine Learning* (VERBRAEKEN *et al*, 2021)

Várias técnicas fundamentais estão envolvidas no atendimento das características de ambientes distribuídos. O aprendizado de máquina distribuído conduz o treinamento sobre um conjunto de dispositivos, chamados *workers*. A técnica de paralelismo tem como seu ponto chave em ML distribuído particionar os dados de treinamento e o parâmetro do modelo. Na era de Big Data, o tamanho de muitos conjuntos de dados pode chegar a centenas de GBs ou até TB. É muito grande para ser armazenado e executado com um único *worker*. O paralelismo de dados é uma estratégia em que os dados de treinamento são particionados em diferentes

partições e aloca cada partição para o *worker* correspondente. Enquanto isso, todos os *workers* têm uma cópia completa do parâmetro do modelo (JIANG *et al.*, 2022).

Um framework de aprendizado de máquina distribuído agrega parâmetros de todos os *workers* após a computação local e atualiza os parâmetros do modelo local de volta nos *workers*. Essa fase de agregação e atualização precisa de um mecanismo de compartilhamento de modelo entre os *workers* para manter a consistência dos parâmetros do modelo. Uma alternativa é manter um parâmetro de modelo global em uma memória distribuída que pode ser acessada por todos os *workers*, chamada de arquitetura *shared-memory*. Nessa arquitetura os *workers* podem ler e gravar dados compartilhados na memória compartilhada por meio de rede de interconexão.

Em comunicação de redes sem fio, as técnicas de aprendizado de máquina distribuído têm sido cada vez mais aplicadas. Os recursos aprimorados dos dispositivos terminais, volume de dados em crescimento explosivo, congestionamento nas interfaces de rádio e preocupação crescente com a privacidade dos dados necessitam do uso dessas técnicas devido ao ambiente exclusivo de redes sem fio (HU *et al*, 2021). Um exemplo de técnica que propõe manter a privacidade dos dados é a chamada Federated Learning, que será explicada no item 2.5.2, a seguir.

2.5.2 *Federated Learning*

Os sistemas de *Federated Learning* (FL) podem ser implantados onde várias partes aprendem conjuntamente uma rede neural profunda precisa, mantendo os próprios dados locais e confidenciais. Existem cenários em que é benéfico ou mesmo obrigatório isolar diferentes subconjuntos de dados de treinamento uns dos outros para manter a privacidade dos dados (VERBRAEKEN *et al*, 2021). Abordagens de FL permitem modelos mais inteligentes, baixa latência, menor consumo de energia e garantia da privacidade. Trabalha sem a necessidade de guardar dados do usuário na nuvem.

Para processar dados de forma segura e robusta, os pesquisadores da Google desenvolveram a abordagem Federated Learning, em 2016. Nessa abordagem, cada dispositivo treina seu modelo de ML de forma local. A atualização dos parâmetros e pesos é enviada para nuvem de forma criptografada. Na nuvem é calculada a média com outras atualizações do usuário, melhorando assim o modelo compartilhado. Os dados permanecem no dispositivo local e nenhum parâmetro individual do dispositivo vai para nuvem (GOOGLE BLOG, 2017). A Figura 14 ilustra como funciona um modelo FL.

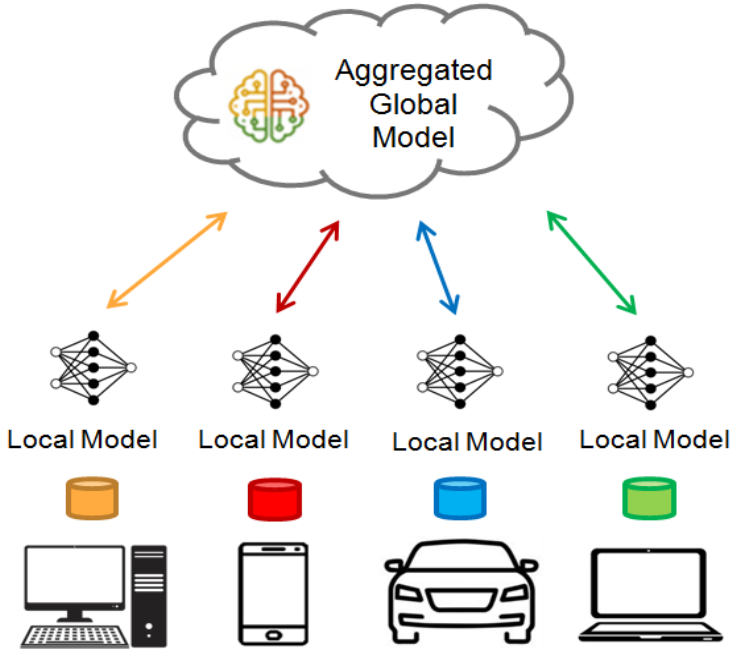


Figura 14: Ilustração do conceito de *Federated Learning*.

Um exemplo de aplicação de FL é a tarefa de previsão da próxima palavra em telefone celular. Para executar essa tarefa, preservando a privacidade dos dados de texto e melhorar a comunicação na rede, é necessário treinar um preditor de forma distribuída, em vez de enviar os dados brutos para um servidor central. Nesta configuração, os dispositivos remotos se comunicam periodicamente com um servidor central para aprender um modelo global. Em cada rodada de comunicação, um subconjunto de telefones selecionados realiza treinamento local em seus dados de usuário distribuídos de forma não idêntica e envia essas atualizações locais ao servidor. Depois de incorporar as atualizações, o servidor envia de volta o novo modelo global para outro subconjunto de dispositivos. Este processo de treinamento iterativo continua em toda a rede até que a convergência seja alcançada (LI *et al*, 2020).

O protocolo de comunicação compreende as etapas de seleção, configuração, relatório, treinamento e agregação para cada rodada. No entanto, essa abordagem possui dados baseados no uso do usuário. Assim, os dados podem sofrer problemas por serem não IID, ter um comportamento desequilibrado e por ser massivamente distribuídos podem apresentar um canal de comunicação limitado. Portanto, a tomada de decisão pode não refletir um cenário global, e o resultado pode divergir quando dispositivos selecionados executam muitas rodadas de atualização local (ABDULRAHMAN et al, 2020).

Na literatura existem vários exemplos de algoritmos de FL e em 2016, McMAHAN *et al.*, pela primeira vez, baseado em paralelismo de dados, propõem um algoritmo *Federated Averaging* (FedAvg). O FedAvg elimina a necessidade de fazer upload dos dados confidenciais dos usuários para um servidor centralizado e possibilita que os dispositivos de

borda treinem um modelo compartilhado localmente em seu próprio conjunto de dados local. Ao agregar as atualizações (gradientes) de modelos locais, o FedAvg atende aos requisitos básicos de proteção de privacidade e segurança de dados.

Os autores afirmam que o FedAvg é capaz de lidar com os dados *non*-IID, mas até certo ponto, pois muitas pesquisas indicaram que uma deterioração no desempenho do FL é quase inevitável com dados *non*-IID. A degradação do desempenho pode ser atribuída principalmente à divergência de peso dos modelos locais resultantes do *non*-IID (ZHU *et al*, 2021). Quando a divergência aumenta, faz com que a convergência se torne mais lenta, piorando o aprendizado.

2.6 Considerações Sobre o Capítulo

Neste capítulo foram apresentados diversos conceitos que são utilizados ao longo deste trabalho. Os conceitos sobre Federated Learning, dados *non*-IID e entropia permitem que o leitor tenha uma visão ampla sobre seus tipos, definições, características, utilização e objetivos. Alguns modelos de FL servem para tratar dados *non*-IID com diversos aspectos, inclusive com entropia. Esses modelos serão apresentados no capítulo seguinte sobre trabalhos relacionados.

3 TRABALHOS RELACIONADOS

Para a elaboração desta dissertação, foi realizada uma revisão sistemática de literatura para identificar trabalhos relacionados, onde foram levantados tópicos como objetivos, questões e estratégia de pesquisa, em torno dos conceitos de *Collaborative Learning* e *Federated Learning* aplicados em ambiente de sistema autônomos distribuídos com dados heterogêneos. Quando as questões de pesquisa foram definidas, foi possível identificar: modelos de *Collaborative Learning* e *Federated Learning* para tomada de decisão, os tipos de algoritmos que estão sendo implementados para decisões preditivas com *Big Data*, as tecnologias e as metodologias realizadas para desenvolver sistemas autônomos de IoT (A-IoT). Os sistemas A-IoT fornecem um ambiente dinâmico e interativo, que detectam o ambiente e tomam decisões de controle para reagir (LEI *et al.*, 2020).

3.1 Revisão Sistemática de Literatura

Para preencher a lacuna no conhecimento sobre a existência e os limites das abordagens de *Collaborative Learning* e *Federated Learning*, realizou-se uma revisão sistemática de literatura (RSL) com uma série de artigos científicos relacionados ao tema deste trabalho, onde algumas questões foram levantadas para a pesquisa e estudo de modelos de CL e FL em *Big Data* sobre sistemas distribuídos e dados *non*-IID, que são dados heterogêneos. A estrutura da RSL é composta de metodologia, questões e estratégia de pesquisa, apresentando resultados e análise comparativa dos modelos de cada publicação pesquisada. No desenvolvimento da RSL adotou-se a metodologia descrita por Kitchenham (KITCHENHAM *et al.*, 2007).

A RSL é formada pela busca de publicações acadêmicas de acordo com um protocolo baseado no processo de mapeamento sistemático, que está estruturado em três fases: (1) planejamento, (2) condução e (3) apresentação dos resultados. A primeira fase contém três etapas: Identificação da necessidade de uma revisão; Definição de questões de pesquisa; Desenvolvimento de um protocolo de revisão. As questões de pesquisa são a parte mais importante de qualquer revisão sistemática conduzindo toda a metodologia de pesquisa (KITCHENHAM *et al.*, 2007). Com base em perguntas de pesquisa e strings de pesquisa, bancos de dados digitais são pesquisados.

Os resultados recuperaram 276 artigos, que incluíram vários formatos (artigos, journals, revistas científicas, etc.), no banco de dados de IEEE Xplore3. Na base de dados IEEE, a pesquisa foi realizada utilizando a pesquisa de metadados padrão. Foi usada a avaliação de qualidade para ponderar a relevância e a importância dos estudos para as questões de pesquisa. A atividade final identificou 30 publicações como as seleções finais, coletando informações necessárias para responder às questões da pesquisa.

3.1.1 Objetivos de Pesquisa

O principal objetivo da revisão sistemática realizada foi identificar estudos primários que relatam modelos de *Collaborative Learning* e *Federated Learning* que têm sido usados para tomada de decisão, usando *Big Data* em um ambiente que utiliza A-IoT.

Os dados em A-IoT podem ser processados e analisados por meio de técnicas de ML, com o objetivo de tomar decisões informadas para controlar as reações dos dispositivos de IoT ao mundo físico. Os dispositivos IoT tornam-se autônomos com a inteligência do ambiente, integrando IoT, ML e controle autônomo.

A abordagem de *Collaborative Learning* tem papel importante para desenvolver algoritmos para tomada de decisão preditiva em um ambiente com A-IoT ou sistemas distribuídos. Essa abordagem combina diferentes redes neurais em uma arquitetura de treinamento paralelo. Além disso, esta revisão busca obter uma visão abrangente dos modelos que vem sendo utilizados em abordagens de *Collaborative Learning* e *Federated Learning*:

- Identificar modelos de *Collaborative Learning* e *Federated Learning* para tomada de decisão de forma colaborativa.
- Identificar quais algoritmos estão sendo implementados para decisões preditivas com *Big Data* e dados *non*-IID.
- Identificar as tecnologias e as metodologias realizadas para desenvolver sistemas com dispositivos autônomos de IoT.

3.1.2 Questões de pesquisa

Para preencher a lacuna no conhecimento sobre a existência e os limites de tal abordagem, algumas questões motivam esta pesquisa sobre modelo de *Collaborative Learning* e *Federated Learning* em *Big Data* e sistema autônomo de IoT. O resumo do estudo com questões formuladas e suas motivações está representado na Tabela 1.

As seguintes questões de pesquisa (Q) abordadas incluem:

Q1 Quais modelos de *Collaborative Learning* ou *Federated Learning* estão sendo utilizados para tomada de decisão?

Q2 Quais são os algoritmos que estão sendo utilizados para decisões preditivas com *Big Data* e dados *non*-IID?

Q3 Quais metodologias realizadas para desenvolver dispositivos IoT autônomos?

Q1 identifica:

- As características dos modelos que têm sido utilizados para tomada de decisão de forma colaborativa.
- Se o artigo possui experimentos que evidenciam o melhor modelo para tomada de decisão.
- Métricas que são utilizadas para avaliação do modelo.

Q2 identifica:

- Tipos de algoritmo que vêm sendo utilizados (Árvore de Decisão, Redes Neurais, Sistema Biológico ou Algoritmo Genético, etc.) em *Big Data* e dados *non*-IID.

Q3 identifica:

- Tipos de dispositivos IoT vêm sendo utilizados.
- Arquiteturas que vêm sendo utilizadas (SOA, PubSub, REST,...) para sistemas autônomos de IoT.

Tabela 1: Questões de pesquisa e motivação

	Questão de Pesquisa	**Motivação**
Q1	Quais modelos de *Collaborative Learning* ou *Federated Learning* estão sendo utilizados para tomada de decisão?	Identificar modelos, os experimentos e métricas de avaliação do modelo de *Collaborative Learning* ou *Federated Learning* para tomada de decisão.
Q2	Quais são os algoritmos que estão sendo utilizados para decisões preditivas com *Big Data* e dados *non*-IID?	Identificar quais tipos algoritmos que estão sendo utilizados (Árvore de Decisão, Redes Neurais, Sistema Biológico ou Genético, etc.) para decisões preditivas com *Big Data* e dados *non*-IID.
Q3	Quais metodologias realizadas para desenvolver dispositivos IoT autônomos?	Identificar os tipos de dispositivos IoT e as arquiteturas utilizadas para desenvolver sistema com dispositivos autônomos de IoT.

3.1.3 Estratégia de pesquisa

Para evitar negligenciar artigos relevantes, foi utilizada uma string de pesquisa refinada alinhada com os objetivos apresentados no item 3.1.1 deste trabalho, que incluiu um grande número de artigos em seus resultados iniciais. Seguimos as orientações (KITCHENHAM *et al.*, 2007), que exigem a identificação dos termos de busca e palavra-chave.

Diferentes grafias foram consideradas ao construir a string de pesquisa usando os operadores booleanos AND e OR. Quando os bancos de dados eram permitidos, a opção de pesquisa avançada inseria a string de pesquisa completa. Para estender o escopo dos resultados, consideramos a população e a intervenção na string de pesquisa. A string de pesquisa foi do seguinte modo:

(("collaborative learning") OR ("federated learning") OR
("federated learning" AND "non-iid") OR
("federated learning" AND "IoT") OR
("federated learning" AND "optimization"))

Strings de busca foram utilizadas para pesquisar em bibliotecas digitais, como por exemplo IEEE, e o número de artigos inicialmente recuperados foram 328. As strings de busca foram ajustadas de acordo com os requisitos dos motores de busca do banco de dados selecionado e foi adicionado filtro por período de publicação entre os anos 2009 e 2023. O número de artigos recuperados reduziram para 279, dentre eles haviam 219 conferências, 54 *journals* e 1 *magazine*. Os artigos a partir do ano 2009 estão relacionados a publicações com conceitos, que foram importantes para contribuir com o capítulo de Fundamentos Teóricos deste trabalho. Dentre os *journals* foram selecionadas 30 publicações mais representativas. A lista das publicações selecionadas e mais relevantes estão disponíveis na Tabela 2, onde é apresentado o nome do modelo de *Federated Learning*, para algumas publicações.

Tabela 2: Publicações selecionadas

Ano	**Autores**	**Título**	**Modelo FL**
2017	McMAHAN, B. *et al.*	*Communication-Efficient Learning of Deep Networks from Decentralized Data*	FedAvg
2019	GAO, X. *et al.*	*iRBP-Motif-PSSM: Identification of RNA-Binding Proteins Based on Collaborative Learning*	
2019	IMANI, M. *et al.*	*A Framework for Collaborative Learning in Secure High-Dimensional Space*	
2019	PENG, Y. *et al.*	*Two-Stream Collaborative Learning With Spatial-Temporal Attention for Video Classification*	
2019	XIE, Y. *et al.*	*Knowledge-based Collaborative Deep Learning for Benign-Malignant Lung Nodule Classification on Chest CT*	
2020	CHEN, Z. *et al.*	*Zero Knowledge Clustering Based Adversarial Mitigation in Heterogeneous Federated Learning*	
2020	CHIU, T-C. *et al.*	*Semisupervised Distributed Learning With Non-IID Data for AIoT Service Platform*	FedSwap
2020	KHOA, T. V. *et al.*	*Collaborative Learning Model for Cyberattack Detection Systems in IoT Industry 4.0*	
2020	MENG, L. *et al.*	*Multi-Colony Collaborative Ant Optimization Algorithm Based on Cooperative Game Mechanism*	
2020	SAHU, A. K. *et al.*	*Federated Optimization in Heterogeneous Networks*	FedProx
2020	WANG, H. *et al.*	*Optimizing Federated Learning on Non-IID Data with Reinforcement Learning*	Favor
2020	WU, C. *et al.*	*Collaborative Learning of Communication Routes in Edge-enabled Multi-access Vehicular Environment*	
2020	ZANNOU, A. *et*	*Relevant node discovery and selection approach for the Internet*	

	al.	*of Things based on neural networks and ant colony optimization*	
2020	ZHAO, L. *et al.*	*Privacy-Preserving Collaborative Deep Learning With Unreliable Participants*	
2021	KARIMIREDDY, S. P. *et al.*	*Stochastic Controlled Averaging for Federated Learning*	SCAFFOLD
2021	REDDI, S. J. *et al.*	*Adaptive Federated Optimization*	FedOpt
2021	ZHANG, W. *et al.*	*Client Selection for Federated Learning With Non-IID Data in Mobile Edge Computing*	CSFedAvg
2022	KIM, J. *et al.*	*A Novel Joint Dataset and Computation Management Scheme for Energy-Efficient Federated Learning in Mobile Edge Computing*	
2022	LI, B. *et al.*	*Federated Anomaly Detection on System Logs for the Internet of Things: A Customizable and Communication-Efficient Approach*	FedLog
2022	LI, B. *et al.*	*Federated End-to-End Learning With Non-IID Data for Vehicular Ad Hoc Networks*	FEEL
2022	LING, Z. *et al.*	*Efficient Device Grouping for Federated Learning Using Maximum Entropy Judgment*	FedEntropy
2022	QIAO, D. *et al.*	*Adaptive Federated Deep Reinforcement Learning for Proactive Content Caching in Edge Computing*	AFPC
2022	ZHANG, J. *et al.*	*Adaptive Federated Learning on Non-IID Data With Resource Constraint*	
2022	ZHANG, Z. *et al.*	*Generative Online Federated Learning Framework for Travel Time Estimation*	GOF-TTE
2022	ZHAO, Z. *et al.*	*Federated Learning With Non-IID Data in Wireless Networks*	
2022	ZHOU, Z. *et al.*	*Cloud-Edge Based Personalized Federated Learning for In-Home Health Monitoring*	FedHome
2023	ITAHARA, S. *et al.*	*Distillation-Based Semi-Supervised Federated Learning for Communication-Efficient Collaborative Training With Non-IID Private Data*	DS-FL
2023	RUBY, R. *et al.*	*Energy-Efficient Multiprocessor-Based Computation and Communication Resource Allocation in Two-Tier Federated Learning Networks*	
2023	YOU, X. *et al.*	*Reschedule Gradients: Temporal Non-IID Resilient Federated Learning*	FedGS
2023	ZHANG, Y. *et al.*	*An Efficient Model Discrepancy-Aware Semi-Asynchronous Clustered Federated Learning Framework*	FedMDS

3.1.4 Resultados da Pesquisa

Este tópico detalha os resultados sistemáticos da Revisão da Literatura. Abordando as questões de pesquisa e apresentando interesses de pesquisa relacionados aos mecanismos de *Collaborative Learning* e *Federated Learning* em *Big Data* sobre sistemas distribuídos. Foram abordadas três questões (Q1, Q2 e Q3), apresentadas na Tabela 1 e neste tópico serão respondidas todas elas com base nas publicações selecionadas listadas na Tabela 2.

A revisão sistemática de literatura procurou características sobre *Collaborative Learning*, *Federated Learning*, *Big Data*, dados *non*-IID e sistemas autônomos distribuídos. Além disso, buscou-se encontrar os tipos de algoritmos utilizados em cada modelo ou

abordagem, que em sua maioria usa o modelo de rede neural CNN. Como métrica a maior parte dos artigos utiliza Acurácia. A Tabela 3 apresenta as publicações e suas características.

Em todas as publicações apresentadas e selecionadas, abordam-se conceitos de *Machine Learning* e *Big Data*, porém em sua maioria não mencionam quais ferramentas de Big Data foram utilizadas para os experimentos, apenas os *datasets* utilizados. A maioria dessas publicações selecionadas, como o trabalho de Brendam McMaham (McMAHAN *et al.*, 2017), usa os conhecidos datasets MNIST e CIFAR-10, em seus experimentos. Alguns autores utilizam o *framework* TensorFlow e a linguagem de programação Python, como por exemplo, os autores de (SHAO *et al.*, 2019). Além de McMaham, autores de publicações mais recentes apresentam modelos de *Federated Learning* para tratar e realizar treinamento com dados *non*-IID. Os trabalhos relacionados recentes tentam melhorar diferentes aspectos, como consumo de energia dos dispositivos (RUBY *et al.*, 2023), privacidade de dados (YOU *et al.*, 2023) e entropia dos dados (ITAHARA *et al.*, 2023) para melhorar a eficácia e a precisão do modelo com base na estrutura básica de FL e no treinamento de dados *non*-IID.

Dentre os trabalhos que tentam melhorar entropia dos dados, os autores de (LING *et al.*, 2022) apresentam um método FL chamado FedEntropy com um esquema dinâmico de agrupamento de dispositivos, que considera a distribuição de dispositivos heterogêneos e contribuições de modelos locais, com base no julgamento de máxima entropia. O trabalho de (ITAHARA *et al.*, 2023) propõe um algoritmo de FL com aprendizado semissupervisionado que aprimora as saídas com média de redução de entropia, para evitar que dados *non*-IID levem os dispositivos móveis a ambiguidade e diminuição da convergência de treinamento, assim os dispositivos trocam essas saídas do modelo agregado, ao invés de trocar parâmetros de modelo, com objetivo de reduzir custo de comunicação.

Nessa revisão sistemática de literatura, obteve-se uma visão geral comparativa, onde percebe-se os problemas em abertos, itens a serem estudados em trabalhos futuros. Alguns problemas em aberto são:

- Gargalos em Eficiência da Comunicação;
- Sistemas heterogêneos e dados *non*-IID;
- Problemas de Privacidade dos dados;
- Dificuldade em prevenir ou detectar ataques maliciosos;
- Limitações de memória dos dispositivos;
- Morosidade na convergência do algoritmo de otimização.

Tabela 3: Publicações e suas características

Publicação	Ano	Autor	Collaborative Learning	Federated Learning	Autonomous IoT	Data Privacy	Energy Consumption	Entropy	Non-IID Data	MNIST dataset	CIFAR-10 dataset	Input Data Type - Text	Input Data Type - Image	Input Data Type - Video	Output Data Type - Text	Output Data Type - Image	Output Data Type - Video	Deep Learning	Q-Learning	Clustering	Classification	Genetic Algorithm	More than one IoT device	Accuracy Metric	Privacy Metric	Connectivity Metric	Error Rate	Convergence rate
Communication-Efficient Learning of Deep Networks from Decentralized Data	2017	McMAHAN, B. *et al.*	X	X		X			X	X	X		X		X			X						X				
iRBP-Motif-PSSM: Identification of RNA-Binding Proteins Based on Collaborative Learning	2019	GAO, X. *et al.*	X									X			X			X						X				
A Framework for Collaborative Learning in Secure High-Dimensional Space	2019	IMANI, M. *et al.*	X			X				X		X	X		X	X		X			X		X	X				
Two-Stream Collaborative Learning With Spatial-Temporal Attention for Video Classification	2019	PENG, Y. *et al.*	X										X	X		X		X			X			X				
Knowledge-based Collaborative Deep Learning for Benign-Malignant Lung Nodule Classification on Chest CT	2019	XIE, Y. *et al.*	X										X		X			X			X			X				
Zero Knowledge Clustering Based Adversarial Mitigation in Heterogeneous Federated Learning	2020	CHEN, Z. *et al.*	X	X		X				X		X			X					X			X					
Semisupervised Distributed Learning With Non-IID Data for AIoT Service Platform	2020	CHIU, T-C. *et al.*	X	X	X	X			X		X		X		X			X						X				
Collaborative Learning Model for Cyberattack Detection Systems in IoT Industry 4.0	2020	KHOA, T. V. *et al.*	X			X						X			X			X			X		X	X	X	X		

Multi-Colony Collaborative Ant Optimization Algorithm Based on Cooperative Game Mechanism	2020	MENG, L. *et al.*	X									X			X							X					X	X
Federated Optimization in Heterogeneous Networks	2020	SAHU, A. K. *et al.*	X	X		X			X	X			X		X			X						X					
Optimizing Federated Learning on Non-IID Data with Reinforcement Learning	2020	WANG, H. *et al.*	X	X		X			X	X	X		X		X				X					X					
Collaborative Learning of Communication Routes in Edge-enabled Multi-access Vehicular Environment	2020	WU, C. *et al.*	X		X							X			X				X				X			X		
Relevant node discovery and selection approach for the Internet of Things based on neural networks and ant colony optimization	2020	ZANNOU, A. *et al.*			X							X			X			X				X		X				
Privacy-Preserving Collaborative Deep Learning With Unreliable Participants	2020	ZHAO, L. *et al.*	X			X				X		X	X		X	X		X						X				
Stochastic Controlled Averaging for Federated Learning	2021	KARIMIREDDY, S. P. *et al.*		X					X	X			X			X		X						X				
Adaptive Federated Optimization	2021	REDDI, S. J. *et al.*		X					X	X	X		X			X		X						X				
Client Selection for Federated Learning With Non-IID Data in Mobile Edge Computing	2021	ZHANG, W. *et al.*		X		X			X	X	X		X		X			X						X				
A Novel Joint Dataset and Computation Management Scheme for Energy-Efficient Federated Learning in Mobile Edge Computing	2022	KIM, J. *et al.*		X			X		X	X			X			X		X						X				
Federated Anomaly Detection on System Logs for the Internet of Things: A Customizable and Communication-Efficient Approach	2022	LI, B. *et al.*		X		X			X			X			X			X						X				
Federated End-to-End Learning With Non-IID Data for Vehicular Ad Hoc Networks	2022	LI, B. *et al.*		X	X				X	X		X	X		X	X		X		X				X			X	

Efficient Device Grouping for Federated Learning Using Maximum Entropy Judgment	2022	LING, Z. *et al.*		X				X	X		X		X			X		X						X		X		
Adaptive Federated Deep Reinforcement Learning for Proactive Content Caching in Edge Computing	2022	QIAO, D. *et al.*		X			X		X					X			X		X					X		X		
Adaptive Federated Learning on Non-IID Data With Resource Constraint	2022	ZHANG, J. *et al.*		X					X	X	X		X			X		X						X				
Generative Online Federated Learning Framework for Travel Time Estimation	2022	ZHANG, Z. *et al.*		X					X			X			X			X									X	
Federated Learning With Non-IID Data in Wireless Networks	2022	ZHAO, Z. *et al.*		X			X		X	X			X			X		X						X		X		
Cloud-Edge Based Personalized Federated Learning for In-Home Health Monitoring	2022	ZHOU, Z. *et al.*		X		X			X			X						X						X				
Distillation-Based Semi-Supervised Federated Learning for Communication-Efficient Collaborative Training With Non-IID Private Data	2023	ITAHARA, S. *et al.*		X				X	X	X			X			X		X						X		X		
Energy-Efficient Multiprocessor-Based Computation and Communication Resource Allocation in Two-Tier Federated Learning Networks	2023	RUBY, R. *et al.*		X			X		X	X			X			X		X						X				
Reschedule Gradients: Temporal Non-IID Resilient Federated Learning	2023	YOU, X. *et al.*		X		X			X	X	X		X			X		X						X				
An Efficient Model Discrepancy-Aware Semi-Asynchronous Clustered Federated Learning Framework	2023	ZHANG, Y. *et al.*		X					X	X			X			X				X				X				

3.2 Modelos de *Collaborative Learning*

O artigo de Shao (SHAO *et al.*, 2019) descreve um sistema de perguntas e respostas (Q&A) e propõe um modelo de seleção de resposta baseado em *Collaborative Learning*, implantando uma arquitetura de treinamento paralela para aprender colaborativamente uma matriz de vetor de palavras com o início da frase por rede neural convolucional e *Long Short Term Memory* (LSTM) bidirecional. LSTM é uma arquitetura de rede neural recorrente (RNN) usada em *Deep Learning* (DL). O modelo consiste em uma parte de aprendizagem colaborativa e uma parte de incorporação de frases não supervisionadas.

Os Métodos de Aprendizado Profundo, também conhecidos como *Deep Learning* são atualmente utilizados em muitas frentes como o reconhecimento facial nas redes sociais, nos carros automatizados e até mesmo em alguns diagnósticos no campo da Medicina. DL permite que modelos computacionais compostos de inúmeras camadas de processamento possam "aprender" representações de dados com diversos níveis de abstração (CHAGAS, 2019).

Zhao (ZHAO *et al.*, 2020) apresenta um sistema de *Collaborative Deep Learning*, chamado SecProbe, que preserva a privacidade e permite aos usuários construir cooperativamente um modelo de DL coletivo com dados de todos os usuários, sem compartilhamento direto de dados. Um usuário treina modelo local com seus próprios dados e depois compartilha apenas os parâmetros do modelo. Os autores realizaram experimentos em uma máquina apenas, ainda é necessário realizar experimento em mais de um nó de *Cluster* ou em mais de um *Cluster*, para aplicar a abordagem de *Collaborative Learning*.

Os autores em (MENG *et al.*, 2020) propõem um algoritmo de otimização de multi-colônia colaborativa de formiga baseado no mecanismo de jogo cooperativo. O artigo apresenta a estratégia de correspondência adaptativa de matriz de feromônio. O modelo de sistema de colônia de formigas usa um algoritmo que seleciona a próxima posição pela regra de proporção aleatória. Depois que todos os caminhos são construídos, os feromônios nos caminhos são atualizados. Com isso, um mecanismo de otimização colaborativa é proposto para melhorar a qualidade da solução. A estratégia é aumentar a concentração de feromônios no caminho comum para acelerar a convergência do algoritmo. A desvantagem do modelo apresentado é que para problemas de grande escala, a eficiência operacional é baixa e demorada. Os autores simularam o experimento em ambiente MatLab, que trata-se de um software interativo voltado para cálculos numéricos, deixando em aberto uma possibilidade de realizar experimentos em um sistema distribuído, para otimizar a eficiência operacional.

O artigo (WU *et al.*, 2020) propõe um esquema de *Collaborative Learning* de roteamento de comunicação para um ambiente de computação de ponta veicular multiacesso.

O esquema proposto emprega um algoritmo de aprendizado de reforço baseado em *Q-Learning* e na colaboração *end-edge-cloud* para encontrar rotas de forma proativa com baixo *overhead* de comunicação. As rotas também são alteradas preventivamente com base nas informações aprendidas.

Khoa *et al.* (2020) propõem um novo sistema de detecção de intrusão baseado em *Collaborative Learning* que implementa uma forma eficiente em grande volume de dados de IoT na Indústria 4.0. Foram desenvolvidos "filtros" inteligentes que podem ser implantados em *gateways* IoT para detectar e prevenir ataques cibernéticos imediatamente. O sistema proposto pode reduzir significativamente a divulgação de informações, bem como o tráfego de rede na troca de dados entre *gateways* IoT. Os autores propõe uma aplicação dos métodos na IoT Industry 4.0, ainda é necessário desenvolver uma solução genérica para implementar em várias aplicações e que seja otimizada em eficiência operacional.

Os autores do artigo de (NORDLUND *et al.*, 2019) apresentam um *framework* de aprendizado cooperativo que permite aos sensores treinar sistemas de DL em seus próprios dados de vídeo locais e percepções compactadas de dados de entrada de sensores vizinhos. Este sistema funde os dados do sensor antes da classificação para permitir que os agentes de aprendizagem lidem com as entradas correlacionadas e cooperem com os sensores vizinhos com custos de comunicação mínimos. Os autores buscam expandir seus algoritmos para incorporar outros elementos de Redes Neurais, a fim de contabilizar o movimento não linear dos alvos e melhorar a precisão dos rastreadores de objetos. Essa melhoria pode ser realizada em um experimento com sistemas distribuídos, usando ferramentas de *Big Data*, buscando aumentar o desempenho e otimizar eficiência operacional.

Os trabalhos citados anteriormente fazem parte de 30 publicações selecionadas na revisão sistemática de literatura, a qual está mais detalhada no item 3.1 deste trabalho. A métrica mais comum dentre essas publicações é o percentual (%) de Acurácia. A Tabela 4 apresenta métricas avaliadas em algumas dessas publicações.

Tabela 4: Publicações e suas métricas

Artigo	**Métrica**
CHEN, Z. *et al.*	Acurácia (%) em *dataset* comum; Acurácia (%) em dados infectados: acurácia de numero de acerto nas 800 amostras, alvo de ataque, usando um conjunto de dados infectados; Taxa de Detecção (DR): taxa de sucesso em que a abordagem de defesa pode sinalizar corretamente os nós maliciosos na rede; Taxa de Detecção Falsa (FDR): taxa que a abordagem de defesa sinaliza os nós normais como nós maliciosos.
CHIU T-C. *et al.*	Acurácia (%)

DU, Z. *et al.*	Conceito, Distribuição de dados, Acurácia (%), Comunicação, Privacidade
ESPOSITO, C. *et al.*	Custo Computacional
GAO, X. *et al.*	Desempenho, Acurácia (%)
HU, W. *et al.*	*Area under roc curve* (AUC), Acurácia (ACC), *Sensitivity* (SEN), *Specificity* (SPF), Precisão (PRC), F1 score (F1).
IMANI, M. *et al.*	Acurácia (%)
KHOA, T. V. *et al.*	Acurácia (%), privacidade, comunicação, *overhead*, tempo de aprendizagem
LEI, L. *et al.*	Gradiente Descendente; Distância de Fischer: é uma medida de dissimilaridade entre duas funções de distribuição de probabilidade (COSTA, 2014).
McMAHAN, H. B. *et al.*	Acurácia (%)
MENG, L. *et al.*	Solução ótima (Opt), a pior solução (Pior), solução média (média), taxa de erro, iteração de convergência (Convergência) e desvio padrão.
MESCHEDER, L. *et al.*	*Volumetric IoU*, Distância Chamfer-L1, *Normal consistency score*
NORDLUND, N. *et al.*	Acurácia (%)
PENG, Y. *et al.*	Acurácia (%)
QIAN, S. *et al.*	Acurácia
QIN, Y. *et al.*	Acurácia geral (OA), acurácia média (AA)
SAHU, A. *et al.*	Acurácia (%)
SEO, J. *et al.*	Acurácia (%)
SHAO, T. *et al.*	*Mean Average Precision* (MAP), *Mean Reciprocal Rank* (MRR), Acurácia (%)
WANG, D. *et al.*	*Mean time to failure* (MTTF), em horas; *Intrusion detection system* (IDS) *interval*, em segundos.
WANG, H. *et al.*	Acurácia (%)
WANG, Q. *et al.*	Acurácia, Eficiência e Eficácia, Tempo e Custo de Computação
WEI, X. *et al.*	*Accumulative reward*; *Average waiting time* (s), em segundos; *Execution time* (s), em segundos.
WU, C. *et al.*	Estabilidade, *Leadership*, Conectividade
XIE, Y. *et al.*	Acurácia (%), F-score, AUC (*area under the receiver operator curve*)
ZANNOU, A. *et al.*	Tempo de execução (s), Acurácia (%), *Recall* (%), F1 score (%), *residual energies* (%)
ZHANG, J. *et al.*	Acurácia (%)
ZHANG, W. *et al.*	Acurácia (%)
ZHAO, L. *et al.*	*Mean relative error* (MRE), Acurácia (%)

3.2.1 Algoritmos

A maioria dos autores das publicações estudadas preferiu utilizar algoritmo de Redes Neurais Convolucionais (CNN) em suas propostas. O artigo (MENG *et al.*, 2020) utiliza o modelo inspirado em formigas, mais conhecido como sistema biológico ou genético. Neste

modelo os caminhos são construídos e o algoritmo seleciona a próxima posição, atualizando os feromônios nesses caminhos. A lógica é aumentar a concentração de feromônios no caminho comum para acelerar a convergência do algoritmo. A Figura 15 demonstra o modelo biológico das funções em uma colônia de formigas.

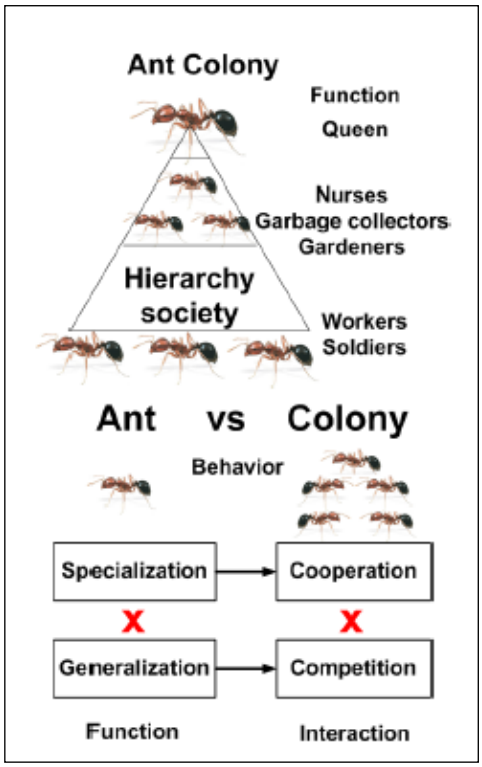


Figura 15: Modelo biológico de formigas (Fonte: Prof. Dr. Julio C. S. Anjos, 2020)

Chen *et al.* (2020) utilizam algoritmo de *clustering*, *k-means*. O artigo de Wu (WU *et al.*, 2020) desenvolveu um esquema que emprega um algoritmo de aprendizado por reforço baseado em *Q-Learning*.

Zhao *et al.* (2020) descrevem com mais detalhes como os algoritmos de SecProbe foram utilizados, pois implementam rede neural da seguinte forma: *perceptron* multicamadas (MLP) com três camadas totalmente conectadas. Para a tarefa de regressão, as funções de ativação da camada oculta e da camada de saída são ReLU e função sigmoide, respectivamente. O número de neurônios na camada oculta é 80. A taxa de aprendizado e o tamanho do *mini-batch* são definidos como 0,01 e 128, respectivamente. Os pesos dos modelos são inicializados aleatoriamente por distribuição normal (com média 0 e desvio padrão 1). A Figura 16 representa o modelo de *Collaborative Deep Learning* e as multicamadas de rede neural implementada pelos autores.

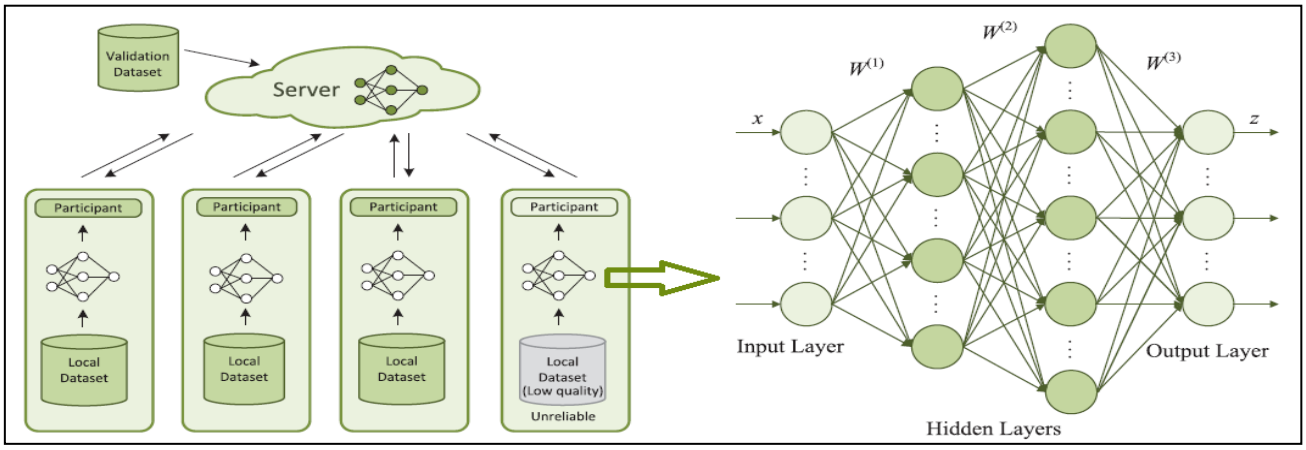


Figura 16: Modelo de *Collaborative Deep Learning* e as multicamadas da rede neural (adaptada de ZHAO *et al.*, 2020)

3.2.2 Alguns Modelos de *Federated Learning*

Federated Learning (FL) é uma abordagem de aprendizado de máquina distribuído que pode atingir o propósito de aprendizado colaborativo de uma grande quantidade de dados que pertencem a diferentes partes, preservando a privacidade desses dados. O artigo (DU *et al.*, 2020) apresenta conceitos teóricos e um breve levantamento dos estudos existentes sobre FL e seu uso em dados de sensores de IoT veicular sem fio. Os autores não apresentam um modelo específico, apenas abordam os desafios técnicos da aplicação de FL em IoT veicular, explicando as melhorias necessárias que devem ser feitas para tecnologias de IoT a fim de oferecer suporte a FL em ambientes veiculares.

Na abordagem Zero-knowledge Clustering (ZeKoC) para FL (CHEN *et al.*, 2020), os autores propuseram atribuir pesos locais às atualizações para filtrar atualizações maliciosas para evitar os ataques de amostra adversários que produzem perturbações durante as etapas de treinamento. O modelo faz uma agregação de parâmetros de ponderação para todos os nós.

Os autores em (McMAHAN *et al*, 2017) apresentam o algoritmo FederatedAveraging (FedAvg). É um método prático para o aprendizado federado de redes profundas, baseado em média de modelo iterativo, sendo realizada uma extensa avaliação empírica, considerando cinco arquiteturas de modelo diferentes e quatro conjuntos de dados. Demonstram em experimentos que a abordagem é robusta para as distribuições de dados desestruturados e *non*-IID, que é uma característica dessa configuração. Os custos de comunicação são a principal restrição, e mostram uma redução nas rodadas de comunicação necessárias de 10 a 100 em comparação com SGD sincronizado. Os detalhes deste algoritmo são apresentados na Figura 17. A partir deste modelo, surgiram outras variantes que serão abordadas a seguir.

Algorithm 4 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
$m \leftarrow \max(C \cdot K, 1)$
$S_t \leftarrow$ (random set of m clients)
for each client $k \in S_t$ **in parallel do**
$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w): *// Run on client k*
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
for batch $b \in \mathcal{B}$ **do**
$w \leftarrow w - \eta \nabla \ell(w; b)$
return w to server

Figura 17: FedAvg (McMAHAN *et al*, 2017)

O artigo de Sahu (SAHU *et al*, 2020) apresenta um framework, FedProx, para lidar com a heterogeneidade em redes federadas. O FedProx pode ser visto como uma generalização e re-parametrização do FedAvg. Embora essa reparametrização faça apenas pequenas modificações no próprio método, essas modificações têm ramificações importantes tanto na teoria quanto na prática. Os autores afirmam que o FedProx permite uma convergência mais robusta do que o FedAvg em conjuntos de dados federados realistas. Em particular, em configurações altamente heterogêneas, o FedProx demonstra um comportamento de convergência significativamente mais estável e preciso em relação ao FedAvg – melhorando a precisão absoluta do teste em 22% em média. A Figura 18 apresenta detalhes desse algoritmo.

FedSwap é proposta como uma nova operação para substituir rodadas parciais de FedAvg na nuvem (CHIU *et al*, 2020). A chave do FedSwap é que, em vez de executar o FedAvg a cada iteração, o FedSwap permite que os dispositivos de borda troquem modelos locais na nuvem. Foi adotada uma estratégia de *roundrobin* para permitir que dois de todos os dispositivos de borda realizem a troca de modelo. Essa operação de troca de modelos entre dispositivos de borda permite que cada modelo tenha uma visão maior de todo o conjunto de dados, portanto, tenha menos divergência de pesos. O objetivo é minimizar frequência de upload e largura de banda. Os detalhes deste algoritmo são apresentados na Figura 19.

Algorithm 5 `FedProx` (Proposed Framework)

Input: $K, T, \mu, \gamma, w^0, N, p_k, k = 1, \cdots, N$
for $t = 0, \cdots, T-1$ **do**
  Server selects a subset S_t of K devices at random (each device k is chosen with probability p_k)
  Server sends w^t to all chosen devices
  Each chosen device $k \in S_t$ finds a w_k^{t+1} which is a γ_k^t-inexact minimizer of: $w_k^{t+1} \approx \arg\min_w h_k(w; w^t) = F_k(w) + \frac{\mu}{2}\|w - w^t\|^2$
  Each device $k \in S_t$ sends w_k^{t+1} back to the server
  Server aggregates the w's as $w^{t+1} = \frac{1}{K}\sum_{k \in S_t} w_k^{t+1}$
end for

Figura 18: FedProx (SAHU *et al*, 2020)

Algorithm 6 Federated Learning With FedSwap

Initialize all clients model with weight w_0;
for $t = 1, 2, ..., T$ **do**
  for *each client* $c = 1, 2, ..., C$ *in parrallel* **do**
    $w_t^c = w_{t-1}^c - \eta \nabla F(w_{t-1}^c)$;
  end
  if $t|k_1 = 0$ *and* $t|k_1k_2 \neq 0$ **then**
    for *each client* $c = 1, 2, ..., C$ **do**
      $w_t^c \leftarrow FedSwap(c, \{w_t^c\}_{c \in C})$;
    end
  end
  if $t|k_1k_2 = 0$ **then**
    $w_t \leftarrow FedAvg(\{w_t^c\}_{c \in C})$;
    for *each client* $c = 1, 2, ..., C$ *in parrallel* **do**
      $w_t^c \leftarrow w_t$;
    end
  end
end
Function `FedSwap` $(c, \{w_t^c\}_{c \in C})$ **:**
  r represent a random client in C;
  $w_t \leftarrow w_t^r$;
  $w_t^r \leftarrow w_t^c$;
  return w_t;
end
Function `FedAvg` $(\{w_t^c\}_{c \in C})$ **:**
  $w_t \leftarrow \sum_{c=1}^{C} \frac{n_c}{n} w_t^c$;
  return w_t;
end

Figura 19: FedSwap (CHIU *et al*, 2020)

Os autores do artigo de Wang H. (WANG H. *et al*, 2020) propõem FAVOR, um framework de controle que escolhe de forma inteligente os dispositivos clientes para participar de cada rodada de aprendizado federado, contrabalançando o viés introduzido por dados *non*-IID e acelerando a convergência. É proposto um mecanismo baseado em *Q-learning* profundo que aprende a selecionar um subconjunto de dispositivos em cada rodada de comunicação para maximizar uma recompensa que incentiva o aumento da precisão da validação e penaliza o uso de mais rodadas de comunicação. Com extensos experimentos realizados no PyTorch, os autores mostram que o número de rodadas de comunicação necessárias no aprendizado federado pode ser reduzido em até 49% no conjunto de dados MNIST, 23% no FashionMNIST e 42% no CIFAR-10, em comparação com o FedAvg.

O principal desafio em FL é realizar treinamento do modelo com dados *non*-IID em dispositivos clientes e o artigo de Zhang (ZHANG *et al*, 2021) propõe um novo algoritmo de FL, chamado CSFedAvg (*Client Selection* FedAvg), para aliviar a degradação da precisão causada por dados *non*-IID. Os clientes com diferentes graus de dados *non*-IID apresentam divergência heterogênea de peso com os clientes detentores de dados IID. No algoritmo, os clientes com menor grau de dados *non*-IID são selecionados para treinar os modelos com maior frequência. A Figura 20 apresenta os detalhes de como funciona esse algoritmo.

Segundo os autores do artigo de Reddi (REDDI *et al*, 2021), modelos federados de otimização, como FedAvg, geralmente são difíceis de ajustar e apresentam um comportamento de convergência desfavorável. Em ambientes não federados, os modelos de otimização adaptativa tiveram um sucesso no combate a esses problemas. O artigo citado propõe um modelo na versão federada de otimização adaptativa, chamado FedOpt, onde é analisada sua convergência na presença de dados heterogêneos para uma configuração não convexa. Os autores realizaram experimentos e mostraram que o uso de otimização adaptativa melhora a acurácia no ambiente de Federated Learning. Usando métodos adaptativos no servidor e algoritmo SGD nos clientes, os autores garantem que seus métodos tenham o mesmo custo de comunicação que o modelo FedAvg e funcionem compartilhando configurações entre dispositivos (REDDI *et al*, 2021). A figura 21 apresenta os passos do algoritmo para o modelo FedOpt.

Algorithm 7 The Design of Client-Selected Federated Averaging (CSFedAvg)

procedure Client-SelectedFederatedAveraging
  Initialize $t = 0$, $\mathbf{w}(t)$, $\mathbf{w}_0(t) \leftarrow \mathbf{w}(t)$, $\mathbb{S} \leftarrow \{\}$, $\mathbb{C}_t \leftarrow \{\}$;
  for each communication round $t = 1, 2, \ldots$ **do**
    $\mathbb{N}_t \leftarrow$ (Randomly select pK clients from $\mathbb{K}$;)
    for each client $c_k \in \mathbb{N}_t \bigcup \mathbb{C}_t \bigcup \mathbb{S}$ parallelly **do**
      $\mathbf{w}_k(t) \leftarrow \text{ClientUpdate}(c_k, \mathbf{w}(t-1))$;
    end for
    if $|\mathbb{S}| < \alpha K$ **then**
      The server updates $\mathbf{w}_0(t)$ using $(\mathbf{w}_0(t-1))$;
      Update $\mathbb{S}$ and $\mathbb{C}_t \longleftarrow$ Algorithm 2;
      $\mathbf{w}(t) \leftarrow \sum_{c_k \in \mathbb{N}_t \bigcup \mathbb{S}} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} \mathbf{w}_k(t)$;
    else
      The server updates $\mathbf{w}_0(t)$ using $(\mathbf{w}(t-1))$;
      $\mathbf{w}(t) \leftarrow \frac{|\mathcal{D}_0|}{|\mathcal{D}|} \mathbf{w}_0(t) + \sum_{c_k \in \mathbb{N}_t \bigcup \mathbb{S}} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} \mathbf{w}_k(t)$;
    end if
  end for
end procedure

function ClientUpdate(c_k, $\mathbf{w}(t-1)$)
  $\mathbf{w}_k(t-1) = \mathbf{w}(t-1)$;
  for each local training epoch from 1 to E **do**
    for batch $b \in B$ **do**
      $\mathbf{w}_k(t) \leftarrow \mathbf{w}(t-1) - \eta_t \nabla F_k(\mathbf{w}(t-1), b)$;
    end for
  end for
  return $\mathbf{w}_k(t)$;
end function

Figura 20: CSFedAvg (ZHANG *et al*, 2021)

Algorithm 8 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 $\nabla F_i(x_{i,k}^t)$
      $x_{i,k+1}^t = \text{CLIENTOPT}(x_{i,k}^t, g_{i,k}^t, \eta_l, t)$
    $\Delta_i^t = x_{i,K}^t - x_t$
  $\Delta_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta_i^t$
  $x_{t+1} = \text{SERVEROPT}(x_t, -\Delta_t, \eta, t)$

Figura 21: FedOpt (REDDI *et al*, 2021)

De acordo com Karimireddy *et al*. (2021), quando os dados são heterogêneos, ou seja, dados *non*-IID, o modelo FedAvg pode sofrer com um *client-drift*, que é uma atualização do modelo local em direção a uma solução local ideal, resultando em instabilidade e convergência lenta do modelo global. O artigo de Karimireddy (KARIMIREDDY *et al*, 2021) propõe um novo modelo com algoritmo de média controlada estocástica, chamado SCAFFOLD, que utiliza variáveis de controle para reduzir a variância e corrigir o *client-drift* em suas atualizações locais. Esse algoritmo requer menos rodadas de comunicação e não é afetado pela heterogeneidade dos dados. Aproveita a similaridade nos dados do cliente, resultando em convergência mais rápida. Segundo os autores, o modelo SCAFFOLD supera FedAvg em experimentos não convexos, em termos de acurácia alcançada. Esse modelo de algoritmo estocástico supera a dissimilaridade de gradiente usando variáveis de controle. A figura 22 demonstra o algoritmo para o modelo proposto.

Algorithm 9 SCAFFOLD: Stochastic Controlled Averaging for federated learning

server input: initial x and c, and global step-size η_g
client i's input: c_i, and local step-size η_l
for each round $r = 1, \ldots, R$ **do**
  sample clients $\mathcal{S} \subseteq \{1, \ldots, N\}$
  communicate (x, c) to all clients $i \in \mathcal{S}$
  on client $i \in \mathcal{S}$ **in parallel do**
    initialize local model $y_i \leftarrow x$
    for $k = 1, \ldots, K$ **do**
      compute mini-batch gradient $g_i(y_i)$
      $y_i \leftarrow y_i - \eta_l \left(g_i(y_i) - c_i + c\right)$
    end for
    $c_i^+ \leftarrow$ (i) $g_i(x)$, or (ii) $c_i - c + \frac{1}{K\eta_l}(x - y_i)$
    communicate $(\Delta y_i, \Delta c_i) \leftarrow (y_i - x, c_i^+ - c_i)$
    $c_i \leftarrow c_i^+$
  end on client
  $(\Delta x, \Delta c) \leftarrow \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} (\Delta y_i, \Delta c_i)$
  $x \leftarrow x + \eta_g \Delta x$ and $c \leftarrow c + \frac{|\mathcal{S}|}{N} \Delta c$
end for

Figura 22: Scaffold (KARIMIREDDY *et al*, 2021)

3.3 Considerações Sobre o Capítulo

Neste capítulo foram apresentados em detalhes os artigos do estado da arte, através de uma revisão sistemática de literatura, que trabalham com modelos de *Federated Learning*. Na revisão de literatura, se obteve uma visão geral dos trabalhos e problemas em aberto. Dentre os problemas em aberto, está o problema de sistemas heterogêneos processarem dados *non*-IID, causando lentidão na convergência de algortimos de otimização.

A partir dos detalhes apresentados e o problema citado, este trabalho propõe um modelo para tratar dados *non*-IID e validar a hipótese de que há relação entre entropia e esse tipo de dados, em um ambiente de sistema distribuído e aprendizado federado. O modelo proposto de *Federated Learning* será abordado e apresentado em detalhes, no capítulo seguinte.

4 MODELO

Este capítulo tem o propósito de descrever o modelo proposto, trabalhando com a hipótese de que a medida de entropia pode ter uma relação direta com dados *non*-IID para otimizar o cálculo do SGD. Este modelo foi apresentado e publicado no artigo de Orlandi (ORLANDI *et al*, 2023).

Os dados *non*-IID são dados que apresentam características de heterogeneidade e possuem alto grau de desordem, ou seja, incerteza de informação encontrada. O cálculo de entropia, por ser uma medida de desordem dos dados, serve para medir essa incerteza no conjunto de dados (GOODFELLOW *et al*, 2017).

Portanto, os dados podem ter menor incerteza com menor entropia. No modelo proposto, o cálculo de entropia é usado como um critério de seleção de dados, comparando com a média de entropia global para definir um novo conjunto de treinamento. Pois como estudado nas seções anteriores, em um algoritmo de Rede Neural temos os dados de treinamento e dados de teste.

Para medir a entropia nos dados *non*-IID, alguns modelos de Federated Learning da literatura foram implementados, principalmente FedAvg, neste trabalho. Com a implementação desses modelos foi possível fazer a devida adaptação com cálculo de entropia e gerar experimentos. O modelo proposto foi elaborado para processar conjuntos de imagens.

4.1 Etapas do Modelo Proposto

O modelo é composto por duas etapas principais. A Figura 23 representa o modelo proposto, onde a primeira etapa é preparação dos dados. Na preparação de dados, a entrada de dados são imagens que serão transformadas em matrizes de índices numéricos. Essas matrizes serão particionadas aleatoriamente para cada cliente em um arquivo com a matriz de índices, os quais são chamados IDX. Esses índices representam as imagens da entrada de dados em formato *non*-IID.

Ainda nessa etapa, é calculada a média de entropia global entre os dados de todos os clientes. Para cada cliente, a entropia de seus dados é calculada. Um total de entropia é obtido entre os clientes e divididos pelo número de clientes existentes dentro do sistema distribuído e federado.

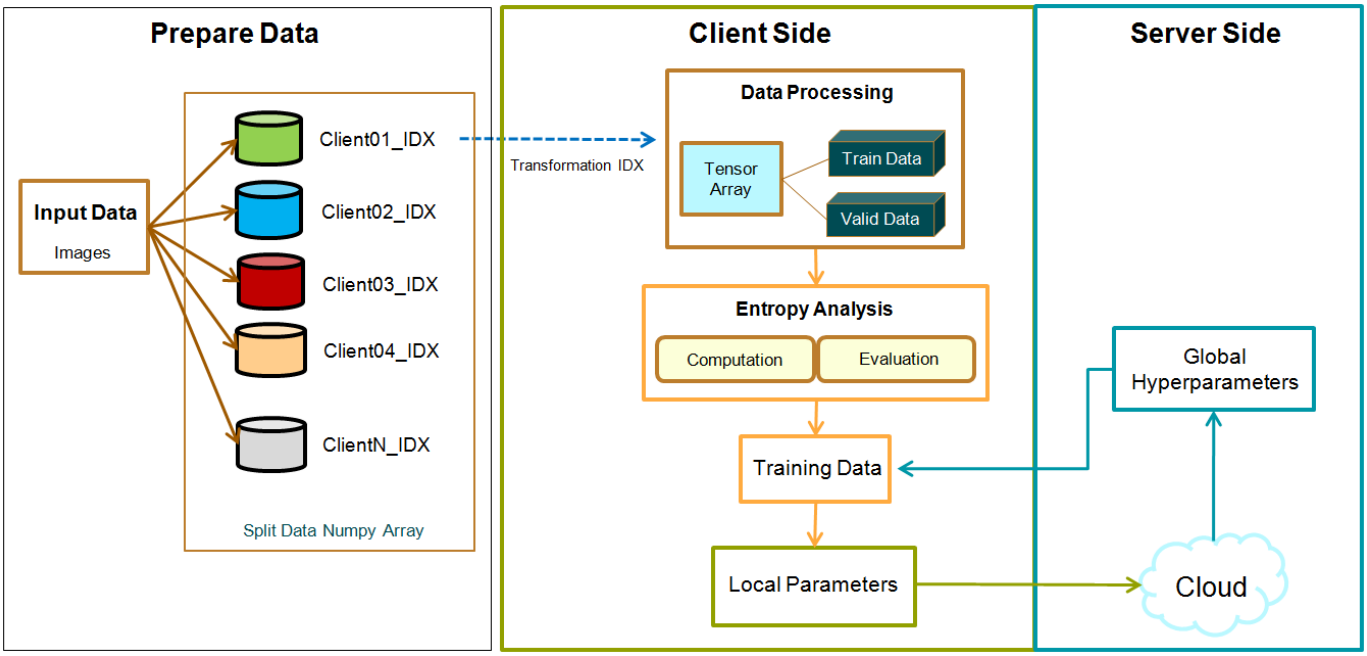


Figura 23: Modelo proposto com análise de entropia no cliente

Na segunda etapa, após a preparação dos dados, no lado do cliente, é realizado o processamento dos dados, onde é realizada uma transformação dos índices IDX em matriz de tensores. Os dados de tensores serão separados em dados de treinamento e dados de validação, na proporção de 80% - 20%. A proporção é uma suposição de IA (GHOLAMY *et al*, 2018) e fornece uma análise de dados consistente do modelo de computação SGD.

Ainda no lado do cliente, é realizado o cálculo da entropia que é comparada com uma entropia média global e avaliada para cada cliente antes da execução do treinamento. Os dados com entropia menor que a média da entropia global são selecionados e treinados localmente. Após o treinamento local, serão obtidos os parâmetros locais, que são enviados para o servidor, em nuvem. No lado do servidor, com os parâmetros recebidos de cada cliente, os hiperparâmetros globais serão calculados e enviados para os clientes, para refazer o treinamento dos dados e obter novamente os parâmetros locais. Esse ciclo se repetirá por épocas e rodadas com quantidades estimadas, conforme o dataset selecionado, e ao término das mesmas, se obterá o resultado do cálculo de acurácia do modelo.

O Algoritmo 10 denominado FedAvg-BE, Federated Averaging Learning com avaliação de entropia de borda, apresenta o modelo desenvolvido para avaliar dados na borda usando entropia. Essa avaliação de dados serve para identificar dados semelhantes com baixa desordem de informação agregada e selecionar os melhores dados no conjunto de dados. Isto fornece informações de aprendizado relevantes para a computação do modelo FL. Este algortimo foi avaliado e definido pelos exaustivos experimentos realizados, que serão descritos nos capítulos de Metodologia e Resultados, desta dissertação.

Primeiramente, o algoritmo inicia com o servidor administrador inicializando o peso global, na sequência calcula a média de entropia dos dados de todos os clientes, com a função *GetMeanEntropy(k)*, na linha 2 do Algoritmo. Entre as linhas 19 e 26, essa função calcula a entropia dos dados de cada cliente e em seguida calcula a média de entropia entre os clientes. Para cada rodada, é determinado o número de clientes participantes, na linha 4.

Para cada cliente selecionado, é realizado o treinamento local e são retornados os pesos otimizados pela função de treinamento *ClientUpdate(k, w)*, na linha 10. Essa função é responsável por criar partições (*batches*) dos dados do cliente, onde existe um número de épocas locais. Para cada época e partição, o modelo é treinado localmente, selecionando os dados do cliente com base no cálculo de entropia, na linha 14, que é representado pela Equação 1 e na linha 29 do Algoritmo. O resultado da entropia da partição é comparado com a média da entropia global, na linha 15. Após retorno de cada cliente, na linha 8, será atualizado o modelo global no administrador. A Figura 24 apresenta a notação adotada para descrição do algoritmo.

Algorithm 10 FedAvg-BE. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate

Server executes:

1: initialize w_0
2: $meanEntropy \leftarrow \text{GetMeanEntropy}(K)$
3: **for** each round t = 1,2,... **do**
4: $m \leftarrow max(C.K, 1)$ ▷ Determine the number of participated clients
5: $S_t \leftarrow$ (random set of m clients)
6: **for** $k \in S_t$ **do**
7: $w_{t+1}^k \leftarrow ClientUpdate(\text{k}, w_t)$ ▷ Get clients improved model
8: $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$ ▷ Update the global model
9:
10: **function** ClientUpdate(k, w) ▷ Run on client k
11: $\beta \leftarrow$ (split P_k into batches of size B)
12: **for** each local epoch i from 1 to E **do**
13: **for** batch $b \in \beta$ **do**
14: $Hbatch \leftarrow \text{Entropy}(b)$ ▷ Calculate entropy for batch data
15: **if** $Hbatch < meanEntropy$ **then**
16: $w \leftarrow w - \eta \nabla \ell(w; b)$ ▷ Do local model training
17: **return** w to server
18:
19: **function** GetMeanEntropy(k) ▷ Calculate Mean Entropy for all clients
20: $meanEntropy \leftarrow 0.0$
21: $totalEntropy \leftarrow 0.0$
22: **for** each client $c \in k$ **do**
23: $Hclient \leftarrow \text{Entropy}(c)$ ▷ Calculate entropy for each client
24: $totalEntropy \leftarrow totalEntropy + Hclient$ ▷ Calculate total entropy
25: $meanEntropy \leftarrow \frac{totalEntropy}{k}$ ▷ Calculate Mean entropy
26: **return** $meanEntropy$
27:
28: **function** Entropy(d) ▷ Calculate Entropy dataset
29: $H \leftarrow -\sum_d p(d) \log_2 p(d)$
30: **return** H

Symbol	Description
w	Weight model parameters
K	Client set
k	Client index
t	Number of round
C	Client Random fraction set
c	Fraction of client set
m	Number of selected clients
S	Random set of m clients
n	Data set size
β	Local batch set
P	Partitioned data
B	Local minibatch size
b	Coefficient from batch set
E	Number of local epochs
η	Learning rate
d	Data set for entropy computing
H	Entropy
p	Fraction probability

Figura 24: Notação adotada para a descrição do algoritmo (ORLANDI *et al.*, 2023)

4.2 Considerações Sobre o Capítulo

Este capítulo apresentou as características da proposta do modelo de Federated Learning e introduziu etapas de funcionamento do mesmo, em um ambiente com dispositivos IoT. O modelo foi desenvolvido para avaliar dados na borda usando entropia, para mitigar o impacto dos dados *non*-IID, em uma arquitetura distribuída e federada. No capítulo seguinte será apresentada a metodologia utilizada para desenvolver o modelo proposto e para produzir os experimentos.

5 METODOLOGIA

Neste capítulo é apresentada a metodologia escolhida para desenvolver o modelo de FL e algoritmo FedAvg-BE propostos, bem como implementar outros modelos de FL da literatura, para produzir experimentos e avaliar seus resultados. As seções seguintes apresentam os itens de configuração para a execução dos experimentos.

Este trabalho é uma pesquisa experimental que busca validar a hipótese de relação entre entropia e dados *non*-IID, pois quanto maior a entropia mais os dados podem ser *non*-IID. Os cenários de experimentação abrangem testar combinação de modelos de FL, *datasets* de imagens, no formato de dados *non*-IID, algoritmos de otimização e configuração de rede neural convolucional.

5.1 Escolhas para a Pesquisa Experimental

Os itens de configuração para os experimentos, que serão apresentados a seguir, foram definidos com base no que foi estudado na revisão sistemática de literatura descrita no capítulo 3, sobre Trabalhos Relacionados, deste trabalho. Alguns softwares que serão mencionados na seção seguinte foram utilizados nos experimentos, com base nos estudos das publicações da revisão de literatura. Além disso, a documentação da empresa NVIDIA recomenda seu *Software Development Kit* (SDK) para ambientes de aprendizado federado, o qual foi desenvolvido com o *framework* PyTorch, também estudado na literatura.

O modelo FedAvg da literatura e seus modelos variantes, que foram utilizados nos experimentos para comparar com o modelo proposto FedAvg-BE, foram escolhidos pela proposta de troca de parâmetros em ambiente distribuído e federado, como também pelos exemplos na documentação do SDK da NVIDIA. Foram utilizados os *datasets* MNIST e CIFAR-10, que são conjuntos de dados de imagens publicamente disponíveis para experimentos de aprendizado de máquina e amplamente utlizados nos trabalhos relacionados. A rede neural escolhida foi a convolucional, CNN, pois é um tipo de rede neural utilizada na literatura para reconhecimento de imagens e processamento de dados em pixel. Os algoritmos de otimização SGD, Adam e RMSprop foram utlizados para testes comparativos, pois as variantes do modelo FedAvg implementam esses algoritmos. Os gráficos foram gerados com software TensorBoard, o qual é recomendado na documentação do SDK da NVIDIA.

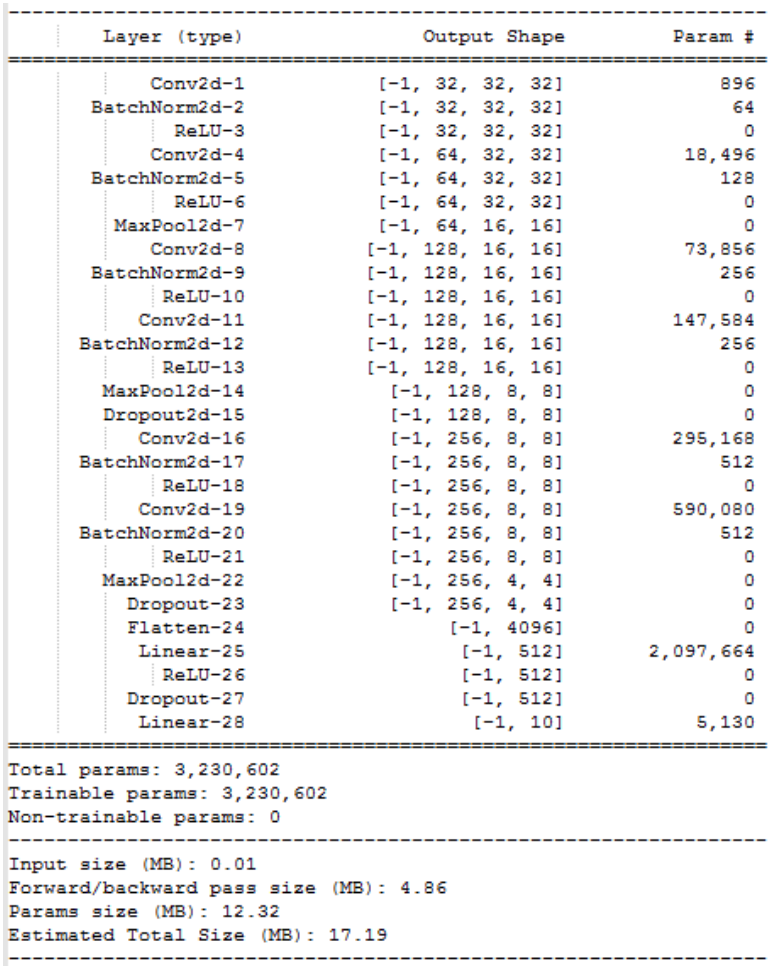
5.2 Configuração para os Experimentos

Neste trabalho foram executados experimentos e os seguintes itens de configuração foram utilizados para gerar os resultados dos experimentos e assim possibilitar comparativos entre os modelos FL com valores de acurácia:

- Modelos:
  - Modelo Centralizado, onde a execução é feita em apenas um cliente, ou seja, não distribuída, para termos de comparação com os modelos seguintes.
  - Quatro (4) modelos de Federated Learning, que foram estudados no capítulo 3 deste trabalho: **FedAvg, FedProx, FedOpt, Scaffold**;
  - O modelo FL proposto neste trabalho: **FedAvg-BE**;
- Dois datasets disponíveis publicamente, que consistem em imagens de 10 classes:
  - **MNIST** – Base de dados de imagens de dígitos manuscritos, no formato 28x28 e pixel do tipo escala de cinza (*GrayScale*). Possui um conjunto de treinamento com 60.000 exemplos e um conjunto de teste com 10.000 exemplos;
  - **CIFAR-10** – Coleção de imagens coloridas de Canadian Institute For Advanced Research, no formato 32x32. Possui um conjunto de treinamento com 50.000 exemplos e um conjunto de teste com 10.000 exemplos;
- Rede Neural: **CNN** – O modelo de rede neural é personalizado para cada dataset. As Figuras 25 e 26 representam os modelos de redes neurais implementados para os dois datasets;

```
----------------------------------------------------------------
      Layer (type)               Output Shape         Param #
================================================================
           Conv2d-1            [-1, 8, 28, 28]              80
             ReLU-2            [-1, 8, 28, 28]               0
      BatchNorm2d-3            [-1, 8, 28, 28]              16
        MaxPool2d-4            [-1, 8, 14, 14]               0
           Conv2d-5           [-1, 16, 14, 14]           1,168
             ReLU-6           [-1, 16, 14, 14]               0
         Dropout2d-7           [-1, 16, 14, 14]               0
      BatchNorm2d-8           [-1, 16, 14, 14]              32
        MaxPool2d-9             [-1, 16, 7, 7]               0
          Conv2d-10             [-1, 32, 7, 7]           4,640
            ReLU-11             [-1, 32, 7, 7]               0
     BatchNorm2d-12             [-1, 32, 7, 7]              64
       MaxPool2d-13             [-1, 32, 3, 3]               0
         Flatten-14                  [-1, 288]               0
          Linear-15                  [-1, 100]          28,900
            ReLU-16                  [-1, 100]               0
         Dropout-17                  [-1, 100]               0
          Linear-18                   [-1, 10]           1,010
            ReLU-19                   [-1, 10]               0
================================================================
Total params: 35,910
Trainable params: 35,910
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.30
Params size (MB): 0.14
Estimated Total Size (MB): 0.44
----------------------------------------------------------------
```

Figura 25: CNN para MNIST



```
----------------------------------------------------------------
      Layer (type)               Output Shape         Param #
================================================================
          Conv2d-1          [-1, 32, 32, 32]             896
     BatchNorm2d-2          [-1, 32, 32, 32]              64
            ReLU-3          [-1, 32, 32, 32]               0
          Conv2d-4          [-1, 64, 32, 32]          18,496
     BatchNorm2d-5          [-1, 64, 32, 32]             128
            ReLU-6          [-1, 64, 32, 32]               0
       MaxPool2d-7          [-1, 64, 16, 16]               0
          Conv2d-8         [-1, 128, 16, 16]          73,856
     BatchNorm2d-9         [-1, 128, 16, 16]             256
           ReLU-10         [-1, 128, 16, 16]               0
         Conv2d-11         [-1, 128, 16, 16]         147,584
    BatchNorm2d-12         [-1, 128, 16, 16]             256
           ReLU-13         [-1, 128, 16, 16]               0
      MaxPool2d-14           [-1, 128, 8, 8]               0
     Dropout2d-15           [-1, 128, 8, 8]               0
         Conv2d-16           [-1, 256, 8, 8]         295,168
    BatchNorm2d-17           [-1, 256, 8, 8]             512
           ReLU-18           [-1, 256, 8, 8]               0
         Conv2d-19           [-1, 256, 8, 8]         590,080
    BatchNorm2d-20           [-1, 256, 8, 8]             512
           ReLU-21           [-1, 256, 8, 8]               0
      MaxPool2d-22           [-1, 256, 4, 4]               0
        Dropout-23           [-1, 256, 4, 4]               0
        Flatten-24                [-1, 4096]               0
         Linear-25                 [-1, 512]       2,097,664
           ReLU-26                 [-1, 512]               0
        Dropout-27                 [-1, 512]               0
         Linear-28                  [-1, 10]           5,130
================================================================
Total params: 3,230,602
Trainable params: 3,230,602
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 4.86
Params size (MB): 12.32
Estimated Total Size (MB): 17.19
----------------------------------------------------------------
```

Figura 26: CNN para CIFAR-10

- Algoritmos de Otimização, estudados no capítulo 2 deste trabalho:
  - SGD;
  - Adam;
  - RMSprop;
- Linguagem de programação: **Python 3.8**
- Software para gerar gráficos: **TensorBoard 2.10.0**
- Framework: **PyTorch 1.13.1 –** Framework que permite executar com GPU e CPU, usando a função chamada “cuda” (PYTORCH, 2022);
- Software Development Kit (SDK): **NVFlare 2.0** – NVIDIA Federated Learning Application Runtime Environment é um SDK extensível, de código aberto e independente de domínio para Aprendizado Federado (NVFLARE, 2022). Este SDK usa conceito de *Job*, que permite executar vários experimentos de forma organizada e ágil. Em cada *Job*, é realizado o deploy da aplicação, em que será executado o experimento;
- Cloud: **Openstack** / CEI;

- Versionamento: **Github** – para versionar e disponibilizar publicamente os códigos-fonte em Python, das aplicações de Federated Learning, utilizados para estudo e experimentos deste trabalho.

A seguir são descritos os passos para implementar a arquitetura de ambiente para Federated Learning e que foi utilizada neste trabalho:

- Na ferramenta Openstack, foram disponibilizadas vinte (20) *Virtual Machines* (VMs) e foram estabelecidas da seguinte forma: um (1) servidor e dezenove (19) clientes na mesma rede. Cada VM terá quatro (4) VCPUs, sistema operacional Linux Ubuntu 18.04.4 LTS e instalação do software de interpretação da linguagem Python 3.8. A definição das VMs será conforme listado na Tabela 5;

Tabela 5: VMs criadas na plataforma Openstack e suas configurações

VM	Nome	RAM	Disco
Server	cei-portoalegre	8 GB	200 GB
site 01	cei-guaiba	4 GB	100 GB
site 02	cei-esteio	4 GB	100 GB
site 03	cei-canoas	4 GB	100 GB
site 04	cei-gravatai	4 GB	100 GB
site 05	cei-viamao	4 GB	100 GB
site 06	cei-cachoeirinha	4 GB	100 GB
site 07	cei-saoleopoldo	4 GB	100 GB
site 08	cei-erechim	4 GB	100 GB
site 09	cei-santarosa	4 GB	100 GB
site 10	cei-cruzalta	4 GB	100 GB
site 11	cei-passofundo	4 GB	100 GB
site 12	cei-cerrolargo	4 GB	100 GB
site 13	cei-trespassos	4 GB	100 GB
site 14	cei-eldoradodosul	4 GB	100 GB
site 15	cei-tapes	4 GB	100 GB
site 17	cei-camaqua	4 GB	100 GB
site 17	cei-pelotas	4 GB	100 GB
site 18	cei-riogrande	4 GB	100 GB
site 19	cei-sapucaiadosul	4 GB	100 GB

- Definição do SDK NVFlare com o framework PyTorch como mecanismo de rede;
- Desenvolvimento de aplicações para cada modelo de Federated Learning, inclusive para o modelo FedAvg-BE.

  A Figura 27 demonstra dois exemplos de como as classes Python foram implementadas em cada aplicação no mecanismo de rede do NVFlare.

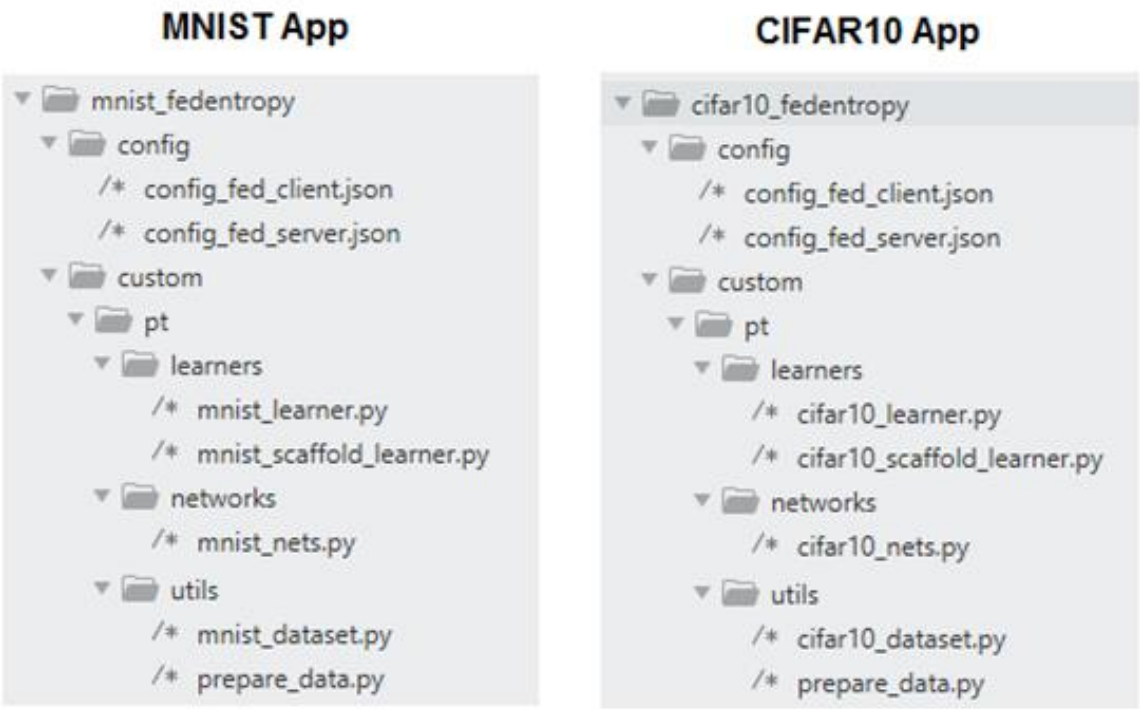


Figura 27: Classes Python para cada aplicação no NVFlare

O arquivo mnist_nets.py, por exemplo, contém as funções para o modelo de rede neural CNN. O arquivo mnist_learner.py contém a classe Learner onde é feita a inicialização das variáveis, normalização dos dados, treinamento e validação. Alguns arquivos úteis como prepare_data.py e mnist_dataset.py foram implementados para preparação dos dados *non*-IID nos clientes, antes de executar o algoritmo de FL.

No seguinte link estão disponíveis os códigos-fonte em python: https://github.com/fernanda-orlandi/dm-nvflare-fedavg-entropy ;

- Estabelecimento de um serviço centralizado para administrar sessões de treinamento. O arquivo config_fed_server.json da aplicação serve para configuração de parâmetros do servidor para executar a aplicação com NVFlare. Nesse arquivo, configura-se o número de rodadas, número de clientes e a classe do modelo de Rede Neural;
- Estabelecimento de um sistema cliente para coordenar os parâmetros do modelo com o serviço central e compartilhar os melhores pesos pelo cálculo da entropia. O arquivo config_fed_client.json da aplicação serve para configuração de parâmetros do cliente para executar a aplicação com NVFlare. Nesse arquivo, configura-se o número de épocas, a taxa de aprendizado e a classe do Learner Executor.

5.3 Considerações Sobre o Capítulo

Este capítulo descreveu a metodologia utilizada para a elaboração dos experimentos, estabelecendo os itens de configuração para produzir os mesmos. Também foram descritos os passos de implementação da arquitura de ambiente para Federated Learning.

Quanto à experimentação, foram executados experimentos distintos, com o objetivo de verificar a efetividade do modelo proposto e comparar os resultados com modelos FL da literatura. Os resultados dos experimentos foram avaliados quanto à acurácia e a redução do tempo de execução. Os diferentes cenários de experimentação e avaliação de seus resultados serão descritos no capítulo seguinte.

6 RESULTADOS E AVALIAÇÃO

Neste capítulo são apresentados os experimentos e seus resultados, como também a avaliação dos resultados experimentais deste trabalho. São também apresentados os comparativos de acurácia e tempo de execução dos modelos de Federated Learning, usando ou não cálculo de entropia, para validar a hipótese e avaliar os resultados obtidos para atingir os objetivos deste trabalho.

Os experimentos foram realizados em 3 fases. Na primeira fase, foram realizados 10 experimentos com base na documentação do SDK NVFlare, para obter o tempo de execução e a acurácia de cada experimento, executando os *Jobs* com 1 servidor e 8 clientes. Ainda nessa fase, 5 testes foram realizados aplicando cálculo de entropia para selecionar os clientes.

Na segunda fase dos experimentos, notou-se que seria melhor selecionar os dados dentro de cada cliente, ao invés de selecionar clientes. Assim seriam selecionados dados com melhor qualidade, através do cálculo de entropia. Nessa fase, houve aumento da quantidade de clientes para 10. Com essas modificações foram realizados mais 70 experimentos.

Na terceira fase de experimentos, houve aumento da quantidade de clientes para 19, conforme o limite de VMs disponibilizadas na plataforma Openstack, para obter mais poder de processamento dos dados *non*-IID. Mais experimentos foram executados, alcançando um total de 115 experimentos realizados neste trabalho, com os datasets e os modelos FL citados no capítulo 5.

6.1 Experimentos e Avaliação de Resultados

Os primeiros experimentos consistem nos exemplos fornecidos pela NVIDIA, conforme documentação do seu SDK NVFlare, com 1 servidor, 8 clientes, 4 épocas, 50 rodadas e algoritmo de otimização SGD. Um dos experimentos de cada dataset é com apenas 1 cliente para obter resultado de acurácia em ambiente centralizado e possibilitar comparar com resultados de um ambiente distribuído. A Tabela 6 representa os primeiros experimentos realizados e seus resultados de acurácia. Com dataset MNIST a acurácia dos experimentos ficou em torno de 98% e para CIFAR-10 a acurácia ficou em torno de 80%.

Tabela 6: Primeiros Experimentos

Job	VMs	Dataset	App	Épocas	Rodadas	Algo. Otimiz.	Learning Rate	Momentum	Acurácia (%)	Tempo Total (min)
6	1 server + 1 client	MNIST	mnist_central	25	1	SGD	0,01	0,9	**99,16**	14
7	1 server + 8 clients	MNIST	mnist_fedavg	4	50	SGD	0,01	0,9	97,89	36
8	1 server + 8 clients	MNIST	mnist_fedprox	4	50	SGD	0,01	0,9	97,75	34
9	1 server + 8 clients	MNIST	mnist_fedopt	4	50	SGD	0,01	0,9	**98,56**	34
10	1 server + 8 clients	MNIST	mnist_scaffold	4	50	SGD	0,01	0,9	**98,18**	37
1	1 server + 1 clients	CIFAR-10	cifar10_central	25	1	SGD	0,01	0,9	**87,88**	120
2	1 server + 8 clients	CIFAR-10	cifar10_fedavg	4	50	SGD	0,01	0,9	75,59	240
3	1 server + 8 clients	CIFAR-10	cifar10_fedprox	4	50	SGD	0,01	0,9	75,78	240
4	1 server + 8 clients	CIFAR-10	cifar10_fedopt	4	50	SGD	0,01	0,9	**79,93**	240
5	1 server + 8 clients	CIFAR-10	cifar10_scaffold	4	50	SGD	0,01	0,9	**82,28**	240

Os experimentos seguintes já incluíram cálculo de entropia no modelo adaptado, oriundo do FedAvg, e os testes foram no âmbito de selecionar os clientes mais aptos conforme a Entropia, porém nesses experimentos os resultados não foram satisfatórios para CIFAR-10, pois a acurácia ficou muito baixa. Para o dataset MNIST a acurácia reduziu, mas a diferença não foi significativa. Na Tabela 7 é possível visualizar os resultados.

Tabela 7: Primeiros experimentos com Entropia e seleção de clientes

Job	VMs	Dataset	App	Épocas	Rodadas	Algo. Otimiz.	Learning Rate	Momentum	Acurácia (%)	Tempo Total (min)	Clients Selected
14	1 server + 8 clients	MNIST	mnist_fedentropy	4	50	SGD	0,01	0,9	83,88	17	2
16	1 server + 8 clients	MNIST	mnist_fedentropy	4	50	SGD	0,01	0,9	93,44	17	5
18	1 server + 8 clients	MNIST	mnist_fedentropy	4	50	SGD	0,01	0,9	93,57	34	5
15	1 server + 8 clients	CIFAR10	cifar10_fedentropy	4	50	SGD	0,01	0,9	24,84	90	3
17	1 server + 8 clients	CIFAR10	cifar10_fedentropy	4	50	SGD	0,01	0,9	25,09	90	6

Após o resultado não satisfatório para CIFAR-10, foi adotada a estratégia de fazer mais experimentos com o dataset MNIST, onde as imagens são mais simples para processar na rede neural, para fazer ajustes e estabilizar o algoritmo do modelo. Além disso, houve aumento da quantidade de clientes para 10, bem como aumento das épocas para 10. Com mais clientes no ambiente, foi necessária nova preparação dos dados *non*-IID para cada cliente e ajustes na classe do modelo de rede neural CNN. Esses ajustes na CNN consistem em alterar o parâmetro *padding* para 1 e incluir a função *Batch Normalization* em cada camada da rede neural. Alguns experimentos foram realizados, modificando o valor do *momentum* entre 0,90 e 0,98. Após essas alterações, primeiramente, os testes foram realizados com modelo FedAvg sem alterações de entropia. Com esses testes, notou-se que seria melhor a seleção de dados dentro de cada cliente, para melhor funcionamento da aplicação no ambiente distribuído com SDK NVflare. Alterações foram feitas para calcular a média de entropia entre todos os clientes e selecionar os dados dentro do método de treinamento de cada cliente, conforme o resultado de entropia fosse menor que a média, assim todos os clientes continuariam executando no ambiente federado e compartilhando seus parâmetros até o fim da execução. A Tabela 8 mostra os resultados de acurácia e tempo de execução, com essas alterações realizadas.

Tabela 8: Experimentos com alterações no cálculo de entropia, em 10 clientes

Job	VMs	Dataset	App	Épocas	Rodadas	Algo. Otimiz.	Learning Rate	Momentum	Acurácia (%)	Tempo Total (min)
20	1 server + 10 clients	MNIST	mnist_fedavg	10	50	SGD	0,01	0,9	**98,83**	65
21	1 server + 10 clients	MNIST	mnist_fedentropy	10	50	SGD	0,01	0,9	**98,83**	60
22	1 server + 10 clients	MNIST	mnist_fedentropy	10	50	SGD	0,01	0,9	98,61	60
23	1 server + 10 clients	MNIST	mnist_fedentropy	10	50	SGD	0,01	0,98	**99,13**	60
27	1 server + 10 clients	MNIST	mnist_fedavg	10	50	SGD	0,01	0,98	98,77	49
28	1 server + 10 clients	MNIST	mnist_fedavg	10	50	SGD	0,01	0,9	98,40	53
29	1 server + 10 clients	MNIST	mnist_fedavg	10	50	SGD	0,01	0,98	**98,92**	52
30	1 server + 10 clients	MNIST	mnist_fedentropy	10	50	SGD	0,01	0,9	**98,81**	53
31	1 server + 10 clients	MNIST	mnist_fedentropy	10	50	SGD	0,01	0,9	98,02	41
32	1 server + 10 clients	CIFAR10	cifar10_fedentropy	10	50	SGD	0,01	0,9	73,58	383

Com os ajustes no modelo de rede neural e no cálculo de entropia, novos experimentos foram realizados para comparar alguns algoritmos de otimização com SDG. Os algoritmos escolhidos para os testes comparativos foram Adam e RMSprop, conforme descritos no capítulo 2 deste trabalho. Para esses dois algoritmos o parâmetro Learning Rate precisou ser modificado para 0,001, enquanto que no SGD foi configurado para 0,01, para o dataset MNIST. Em alguns experimentos, usou-se 40 épocas, mas essa alteração de 10 épocas para 40 apresentou leve redução na acurácia, então a opção foi manter as 10 épocas para os próximos experimentos. Todos os experimentos foram realizados também de forma a comparar os resultados do modelo FedAvg com e sem cálculo de entropia. A Tabela 9 apresenta os resultados dos experimentos comparativos entre os algoritmos de otimização e algumas variações de parâmetros para o dataset MNIST.

Tabela 9: Experimentos com o dataset MNIST, em 10 clientes

Job	VMs	Dataset	App	Épocas	Rodadas	Algo. Otimiz.	Learning Rate	Momentum	Acurácia (%)	Tempo Total (min)
33	1 server + 10 clients	MNIST	mnist_fedentropy	10	50	Adam	0,001		**94,76**	41
34	1 server + 10 clients	MNIST	mnist_fedentropy	40	50	Adam	0,001		94,28	120
36	1 server + 10 clients	MNIST	mnist_fedentropy	10	50	RMSprop	0,001		90,92	41
37	1 server + 10 clients	MNIST	mnist_fedentropy	10	50	RMSprop	0,001	0,9	**93,27**	42
38	1 server + 10 clients	MNIST	mnist_fedentropy	40	50	RMSprop	0,001	0,9	**93,64**	262
39	1 server + 10 clients	MNIST	mnist_fedavg	10	50	RMSprop	0,001	0,9	**97,41**	60
40	1 server + 10 clients	MNIST	mnist_fedavg	40	50	RMSprop	0,001	0,9	**97,17**	180
41	1 server + 10 clients	MNIST	mnist_fedavg	10	50	Adam	0,001		**97,73**	55
42	1 server + 10 clients	MNIST	mnist_fedavg	40	50	Adam	0,001		97,31	180
50	1 server + 10 clients	MNIST	mnist_fedentropy	40	50	SGD	0,01	0,9	**98,18**	46
60	1 server + 10 clients	MNIST	mnist_fedentropy	10	50	SGD	0,01	0,9	98,01	53

Para o dataset CIFAR-10 também foram realizados vários experimentos com os modelos FedAvg e FedAvg-BE, executando com a variação dos algoritmos de otimização SGD, Adam e RMSprop, conforme demonstra na Tabela 10. Alguns experimentos com algoritmo de otimização RMSprop apresentaram baixa acurácia. Isso ocorreu devido ao

parâmetro *Momentum* e discrepâncias na rede neural para esse tipo de algoritmo. Com isso, foi necessário realizar ajustes, como alterar a quantidade de camadas do modelo de rede neural CNN, entre 3 e 6 camadas, bem como remover o parâmetro *Momentum*.

O ajuste para o modelo de rede neural foi mais complexo para o dataset CIFAR-10, com isso foram necessários cerca de 40 testes até obter uma rede neural mais adequada para este dataset. O processamento de imagens do dataset CIFAR-10 exige um modelo de rede neural CNN com 3 blocos e cada bloco duas camadas convolucionais, sendo uma delas a camada residual, o que exige um tempo de processamento maior, mas proporciona maior acurácia, conforme sugestão na documentação do SDK da NVIDIA (NVFLARE, 2022).

Houve melhor convergência quando se alterou o parâmetro de taxa de aprendizagem (*Leaning Rate*) para 0,001 na execução com o algoritmo SGD e para 0,0001 na execução com os algoritmos Adam e RMSprop. O uso do algoritmo SGD apresentou melhores resultados de acurácia para o dataset CIFAR-10, com e sem cálculo de entropia.

Tabela 10: Experimentos com o dataset CIFAR-10, em 10 clientes

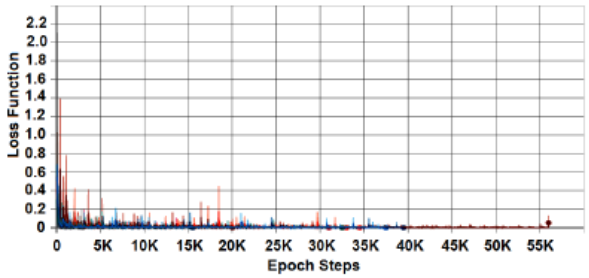
Job	VMs	Dataset	App	Épocas	Rodadas	Algo. Otimiz.	Learning Rate	Momentum	Acurácia (%)	Tempo Total (min)
43	1 server + 10 clients	CIFAR10	cifar10_fedavg	10	50	SGD	0,01	0,9	**84,03**	480
44	1 server + 10 clients	CIFAR10	cifar10_fedavg	10	50	Adam	0,001		**73,18**	518
48	1 server + 10 clients	CIFAR10	cifar10_fedentropy	10	50	Adam	0,001		56,29	360
51	1 server + 10 clients	CIFAR10	cifar10_fedavg	10	50	SGD	0,01	0,9	**85,36**	540
52	1 server + 10 clients	CIFAR10	cifar10_fedavg	10	50	Adam	0,001		64,67	566
53	1 server + 10 clients	CIFAR10	cifar10_fedavg	10	50	RMSprop	0,001	0,9	26,03	530
54	1 server + 10 clients	CIFAR10	cifar10_fedentropy	10	50	SGD	0,01	0,9	**80,74**	400
55	1 server + 10 clients	CIFAR10	cifar10_fedentropy	10	50	Adam	0,001		40,46	420
57	1 server + 10 clients	CIFAR10	cifar10_fedavg	10	50	RMSprop	0,001	0,9	31,33	142
58	1 server + 10 clients	CIFAR10	cifar10_fedavg	10	50	Adam	0,001		59,23	139
59	1 server + 10 clients	CIFAR10	cifar10_fedavg	10	50	SGD	0,01	0,9	**79,92**	137
61	1 server + 10 clients	CIFAR10	cifar10_fedavg	10	50	SGD	0,01	0,9	65,50	88
62	1 server + 10 clients	CIFAR10	cifar10_fedavg	10	50	SGD	0,01	0,9	58,17	52
63	1 server + 10 clients	CIFAR10	cifar10_fedavg	10	50	SGD	0,01	0,9	**76,90**	474

64	1 server + 10 clients	CIFAR10	cifar10_fedavg	30	50	SGD	0,001	0,9	68,79	300
65	1 server + 10 clients	CIFAR10	cifar10_fedavg	10	50	SGD	0,001	0,9	**75,98**	139
66	1 server + 10 clients	CIFAR10	cifar10_fedavg	10	50	RMSprop	0,001	0,9	30,29	136
67	1 server + 10 clients	CIFAR10	cifar10_fedavg	10	50	RMSprop	0,001		59,97	135
68	1 server + 10 clients	CIFAR10	cifar10_fedavg	10	50	RMSprop	0,001		47,51	120
69	1 server + 10 clients	CIFAR10	cifar10_fedavg	10	50	RMSprop	0,0001		48,27	135
70	1 server + 10 clients	CIFAR10	cifar10_fedavg	30	50	RMSprop	0,0001		59,05	360
71	1 server + 10 clients	CIFAR10	cifar10_fedavg	10	50	Adam	0,0001		55,17	137
72	1 server + 10 clients	CIFAR10	cifar10_fedavg	30	50	Adam	0,0001		68,99	400
73	1 server + 10 clients	CIFAR10	cifar10_fedavg	10	50	SGD	0,001	0,9	69,31	147
74	1 server + 10 clients	CIFAR10	cifar10_fedavg	30	50	SGD	0,001	0,9	**75,65**	394
75	1 server + 10 clients	CIFAR10	cifar10_fedavg	10	50	SGD	0,001	0,9	74,12	96
76	1 server + 10 clients	CIFAR10	cifar10_fedavg	30	50	SGD	0,001	0,9	**79,06**	428
77	1 server + 10 clients	CIFAR10	cifar10_fedavg	10	50	SGD	0,001	0,9	**81,91**	579
78	1 server + 10 clients	CIFAR10	cifar10_fedavg	10	50	RMSprop	0,0001		60,18	526
79	1 server + 10 clients	CIFAR10	cifar10_fedavg	10	50	Adam	0,0001		74,17	540
80	1 server + 10 clients	CIFAR10	cifar10_fedavg	10	50	SGD	0,001	0,9	**81,78**	495
81	1 server + 10 clients	CIFAR10	cifar10_fedentropy	10	50	SGD	0,001	0,9	**74,98**	393
82	1 server + 10 clients	CIFAR10	cifar10_fedentropy	10	50	Adam	0,0001		58,88	399
83	1 server + 10 clients	CIFAR10	cifar10_fedentropy	10	50	RMSprop	0,0001		53,22	396

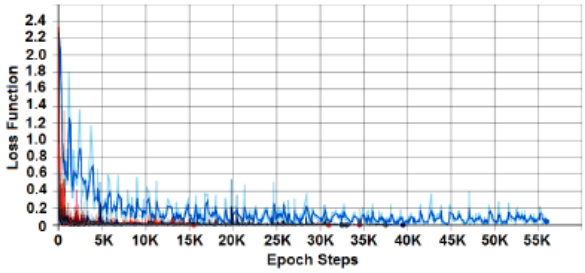
As Figuras 28, 29, 30, 31, 32 e 33 apresentam comparativos de resultados dos experimentos com o modelo FedAvg e o modelo FedAvg-BE. As Figuras 28, 29 e 30 representam os resultados para o dataset MNIST e as Figuras 31, 32 e 33 para o dataset CIFAR-10. Esses experimentos foram executados com a configuração de dez (10) épocas e cinquenta (50) rodadas. Os dados *non*-IID, por serem heterogêneos, são responsáveis pela alta oscilação nos resultados da função de perda.

Nos gráficos das Figuras com *Loss Function* e *Global Accuracy* é apresentado comparativo de resultados do modelo FedAvg e do modelo proposto FedAvg-BE. Nos gráficos de *Loss Function*, cada linha colorida representa um dispositivo cliente. Nos gráficos de *Global Accuracy*, existem duas linhas. Uma se refere à convergência do algoritmo e a outra se refere a uma suavização da ferramenta TensorBoard. Quando aplicado o cálculo de entropia é possível observar nos gráficos, que ocorre oscilação na convergência do algoritmo de otimização e na função de perda. Isso era algo esperado, pois menos dados são processados no treinamento, independente do tipo de conjunto de dados.

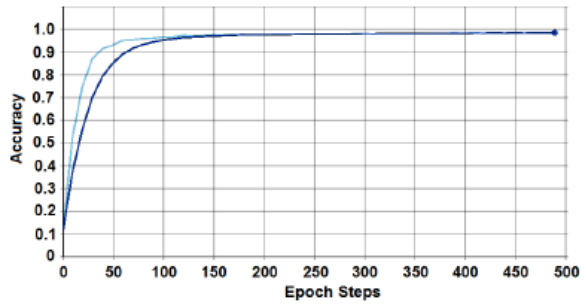
Ao comparar os algoritmos de otimização SDG, Adam e RMSprop, nota-se que o algoritmo SGD obteve melhor acurácia, tanto para o dataset MNIST quanto para o dataset CIFAR-10, considerando os dados *non*-IID em cada cliente.



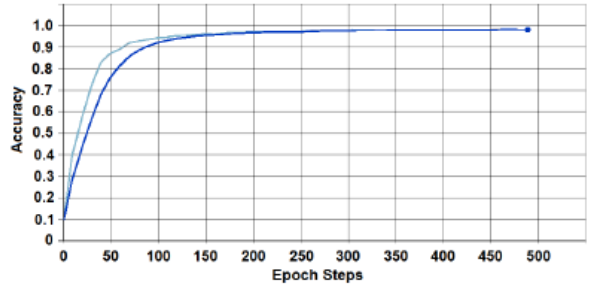
(a) Job 28 - Loss Function - FedAvg



(c) Job 31 - Loss Function - FedAvg-BE

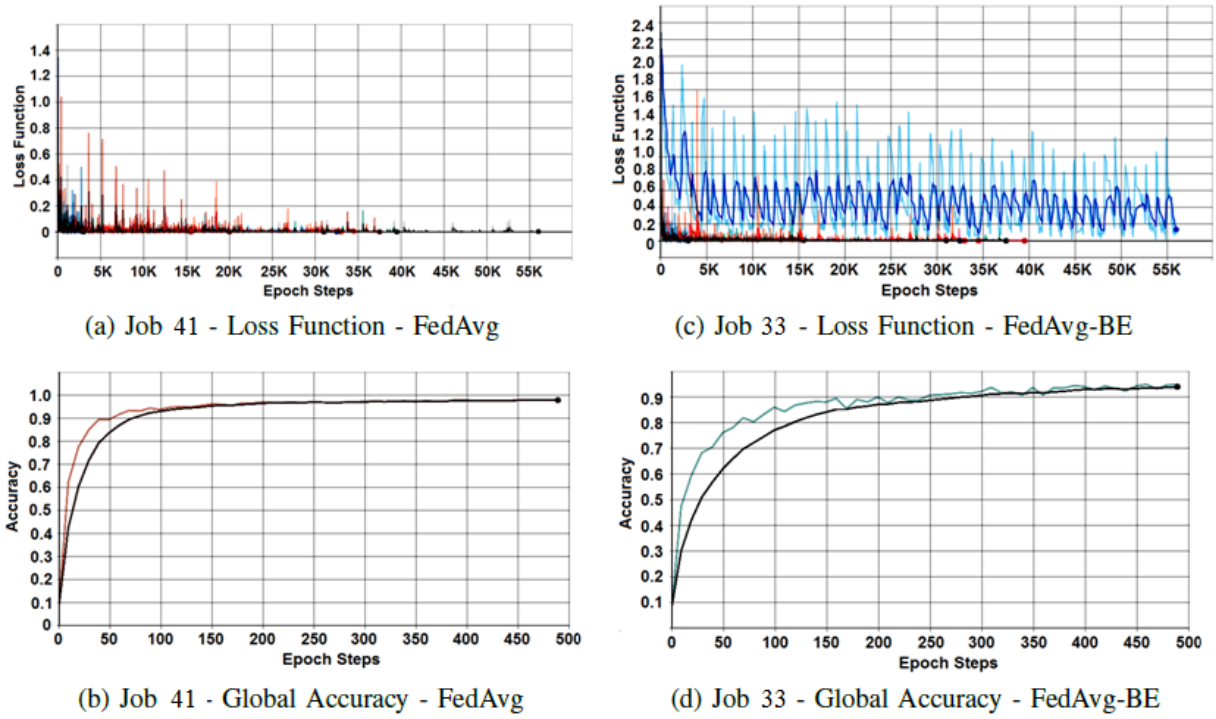


(b) Job 28 - Global Accuracy - FedAvg



(d) Job 31 - Global Accuracy - FedAvg-BE

Figura 28: Gráficos de resultados para o dataset MNIST com SGD em 10 clientes, comparando o modelo FedAvg com o modelo proposto FedAvg-BE



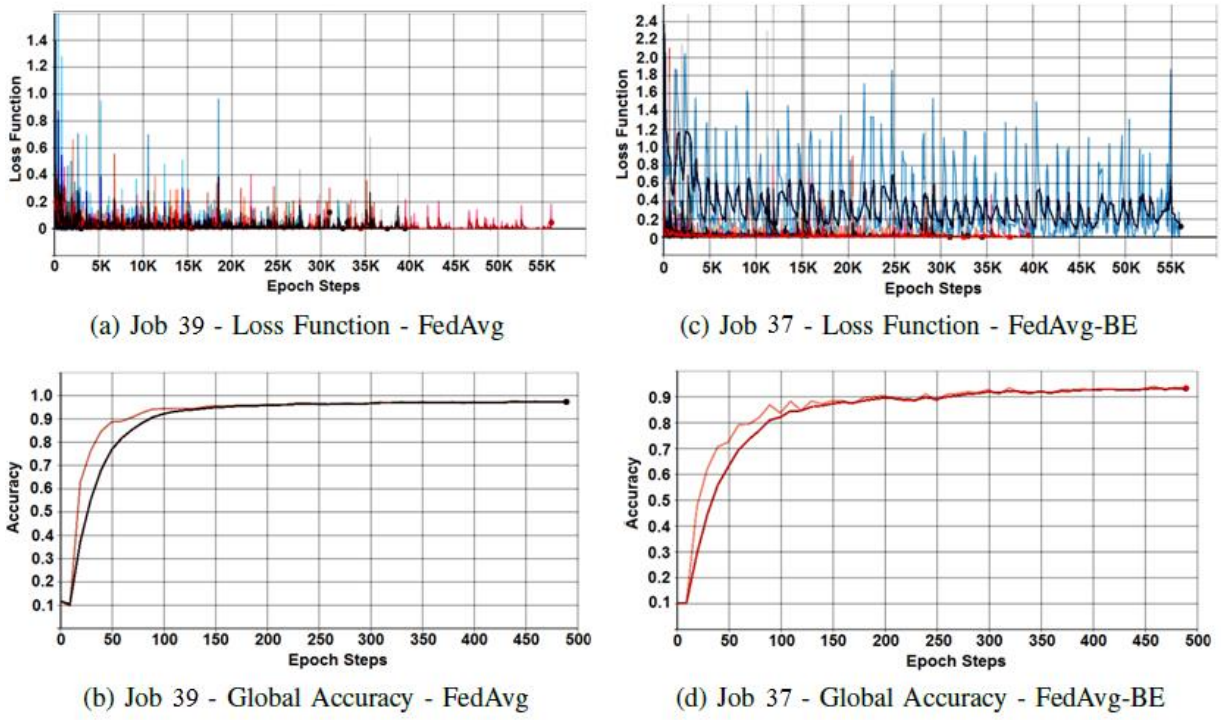
(a) Job 41 - Loss Function - FedAvg

(c) Job 33 - Loss Function - FedAvg-BE

(b) Job 41 - Global Accuracy - FedAvg

(d) Job 33 - Global Accuracy - FedAvg-BE

Figura 29: Gráficos de resultados para o dataset MNIST com Adam em 10 clientes, comparando o modelo FedAvg com o modelo proposto FedAvg-BE



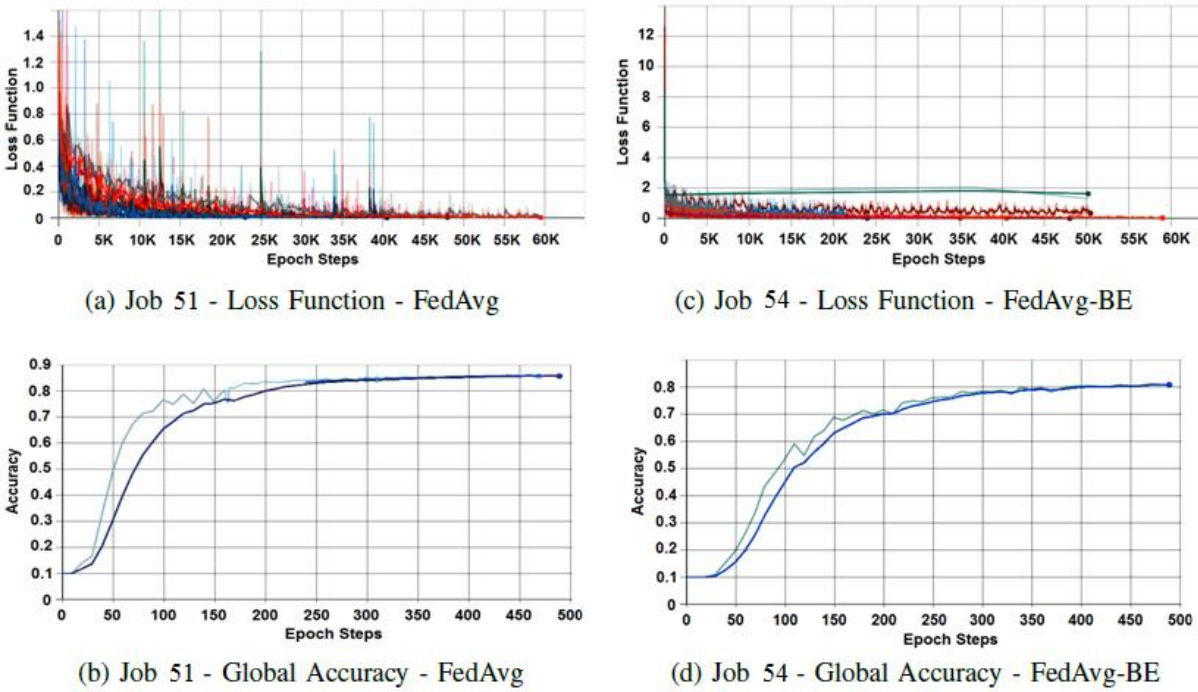
(a) Job 39 - Loss Function - FedAvg

(c) Job 37 - Loss Function - FedAvg-BE

(b) Job 39 - Global Accuracy - FedAvg

(d) Job 37 - Global Accuracy - FedAvg-BE

Figura 30: Gráficos de resultados para o dataset MNIST com RMSprop em 10 clientes, comparando o modelo FedAvg com o modelo proposto FedAvg-BE



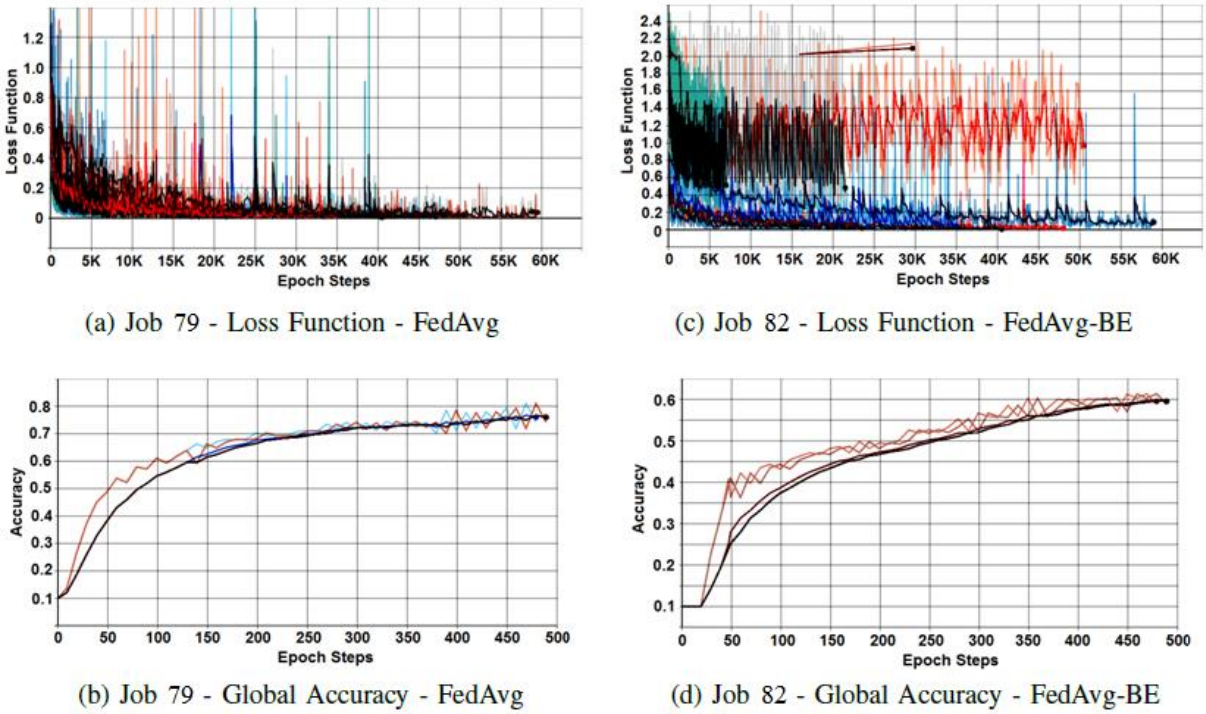
(a) Job 51 - Loss Function - FedAvg

(b) Job 51 - Global Accuracy - FedAvg

(c) Job 54 - Loss Function - FedAvg-BE

(d) Job 54 - Global Accuracy - FedAvg-BE

Figura 31: Gráficos de resultados para o dataset CIFAR-10 com SGD em 10 clientes, comparando o modelo FedAvg com o modelo proposto FedAvg-BE



(a) Job 79 - Loss Function - FedAvg

(b) Job 79 - Global Accuracy - FedAvg

(c) Job 82 - Loss Function - FedAvg-BE

(d) Job 82 - Global Accuracy - FedAvg-BE

Figura 32: Gráficos de resultados para o dataset CIFAR-10 com Adam em 10 clientes, comparando o modelo FedAvg com o modelo proposto FedAvg-BE

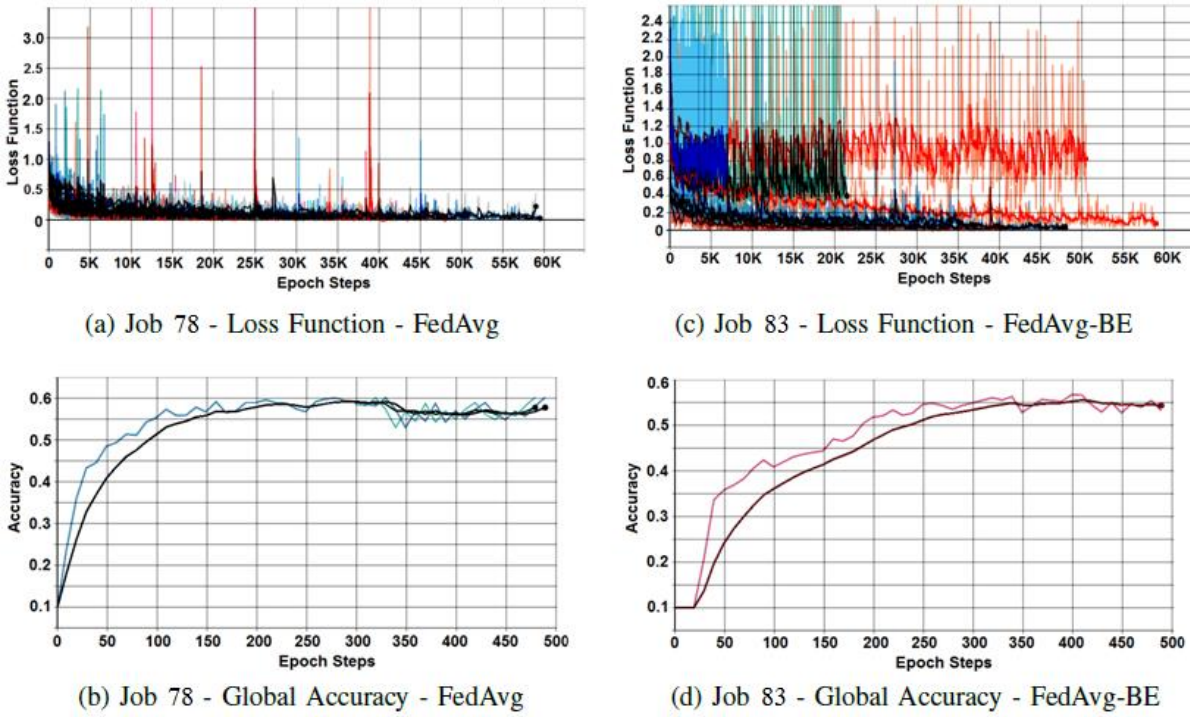


Figura 33: Gráficos de resultados para o dataset CIFAR-10 com RMSprop em 10 clientes, comparando o modelo FedAvg com o modelo proposto FedAvg-BE

A Tabela 11 apresenta a variação dos resultados obtidos com o algoritmo SGD, que mostrou melhor resultado em acurácia, tanto para execução do modelo FedAvg quanto para o modelo proposto, que inclui cálculo de entropia, FedAvg-BE. Os *Jobs* 28 e 31 do dataset MNIST e os *Jobs* 51 e 54 do dataset CIFAR-10 são os melhores casos obtidos dos experimentos, considerando a estabilidade da convergência do SGD com a variável *Momentum* em 0,9 e o uso da função *Batch Normalization*. Ao observar esses *Jobs*, nota-se que para o dataset MNIST ocorreu redução do tempo de execução em 12 minutos, que representa uma redução em torno de 22%, e redução de menos de 1% para a acurácia. Para o conjunto de dados CIFAR-10 ocorreu redução do tempo de execução em torno de 2 horas, que representa uma redução em torno de 26%, e redução entre 4% e 5% para a acurácia.

Os gráficos das Figuras 28 e 31 mostram que houve uma leve diferença na convergência do algoritmo de otimização SGD e na função de perda, devido à seleção de dados, quando aplicado o cálculo de entropia. Com isso, é possível comparar os resultados dos experimentos com o modelo FedAvg e com o modelo proposto FedAvg-BE, para selecionar dados com menor entropia.

Tabela 11: Comparativo da variação dos resultados para os datasets MNIST e CIFAR-10 usando SGD, em 10 clientes

Dataset	Métrica	Resultado FedAvg	Resultado FedAvg-BE	Notas
MNIST	Job	28	31	
	Acurácia	98,40%	98,02%	↓ 0,38%
	Tempo de Execução	53 min	41 min	↓ 12 min 22%
CIFAR-10	Job	51	54	
	Acurácia	85,36%	80,74%	↓ 4,62%
	Tempo de Execução	540 min	400 min	↓ 140 min 26%

Houve aumento da quantidade de clientes para 19, mantendo-se 10 épocas e 50 rodadas. Com mais clientes no ambiente, foi necessária nova preparação dos dados *non*-IID para cada cliente e ajustes na classe do modelo de rede neural CNN. Esses ajustes na CNN consistem em incluir a função de *Batch Normalization* em cada camada da rede neural e alterar a função *Dropout* para algumas das camadas. Após essas alterações, os testes foram realizados com o modelo FedAvg sem alterações de entropia e depois com o modelo FedAvg-BE para cada algoritmo de otimização. A Tabela 12 lista a sequência de experimentos realizados com 19 clientes, para os datasets MNIST e CIFAR-10.

Tabela 12: Experimentos com os datasets MNIST e CIFAR-10, em 19 clientes

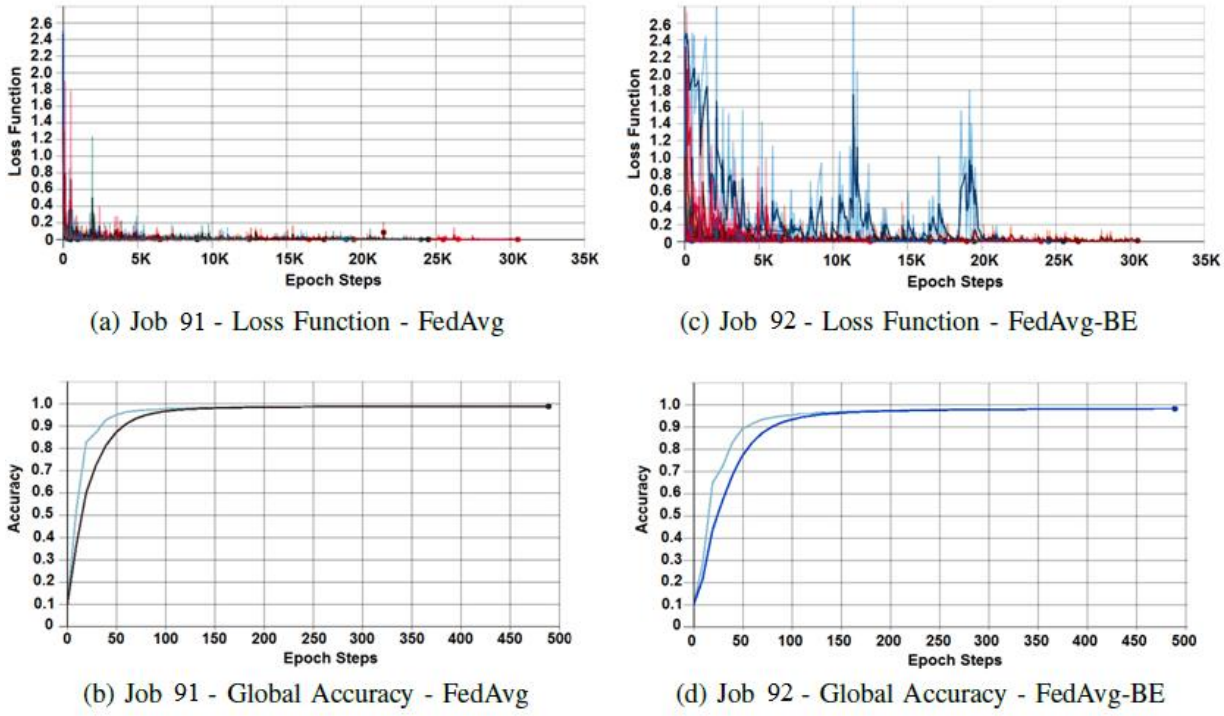
Job	VMs	Dataset	App	Épocas	Rodadas	Algo. Otimiz.	Learning Rate	Momentum	Acurácia (%)	Tempo Total (min)
91	1 server + 19 clients	MNIST	mnist_fedavg	10	50	SGD	0,01	0,9	**98,88**	39
92	1 server + 19 clients	MNIST	mnist_fedentropy	10	50	SGD	0,01	0,9	**98,31**	34
97	1 server + 19 clients	MNIST	mnist_fedavg	10	50	Adam	0,001		**98,43**	41
98	1 server + 19 clients	MNIST	mnist_fedentropy	10	50	Adam	0,001		**98,08**	35
99	1 server + 19 clients	MNIST	mnist_fedavg	10	50	RMSprop	0,001	0,9	**98,19**	42
100	1 server + 19 clients	MNIST	mnist_fedentropy	10	50	RMSprop	0,001	0,9	93,01	36
106	1 server +	CIFAR10	cifar10_fedavg	10	50	SGD	0,01	0,9	**83,47**	380

	19 clients									
107	1 server + 19 clients	CIFAR10	cifar10_fedentropy	10	50	SGD	0,01	0,9	78,68	350
108	1 server + 19 clients	CIFAR10	cifar10_fedavg	10	50	Adam	0,001		**77,81**	393
109	1 server + 19 clients	CIFAR10	cifar10_fedentropy	10	50	Adam	0,001		71,99	375
110	1 server + 19 clients	CIFAR10	cifar10_fedavg	10	50	RMSprop	0,001		**70,21**	460
111	1 server + 19 clients	CIFAR10	cifar10_fedentropy	10	50	RMSprop	0,001		60,68	387

Considerando os 19 clientes, as Figuras 34, 35, 36, 37, 38 e 39 apresentam comparativos de resultados dos experimentos com o modelo FedAvg e o modelo FedAvg-BE. As Figuras 34, 35 e 36 representam resultados para o dataset MNIST e as Figuras 37, 38 e 39 para o dataset CIFAR-10. Esses experimentos foram executados mantendo a configuração de 10 épocas e 50 rodadas. Os dados *non*-IID, por serem heterogêneos, são responsáveis pela oscilação nos resultados da função de perda.

Nos gráficos de *Loss Function* e *Global Accuracy* são comparados os resultados do modelo FedAvg com o modelo FedAvg-BE. Nos gráficos de *Loss Function*, cada linha colorida representa um dispositivo cliente. Nos gráficos de *Global Accuracy*, existem duas linhas. Uma se refere à convergência do algoritmo e a outra se refere a uma suavização da ferramenta TensorBoard. Quando aplicado o cálculo de entropia é possível observar nos gráficos, que ocorre oscilação na convergência do algoritmo de otimização e na função de perda. Esse comportamento era algo esperado, pois menos dados são processados no treinamento, independente do tipo de conjunto de dados. O aumento de número de clientes não afetou os resultados, sendo estes semelhantes aos resultados obtidos com 10 clientes e com os dois modelos FedAvg e FedAvg-BE.

Ao comparar os algoritmos de otimização SDG, Adam e RMSprop, nota-se que o algoritmo SGD obteve melhor acurácia, tanto para o dataset MNIST quanto para dataset CIFAR-10, considerando os dados *non*-IID em cada cliente.



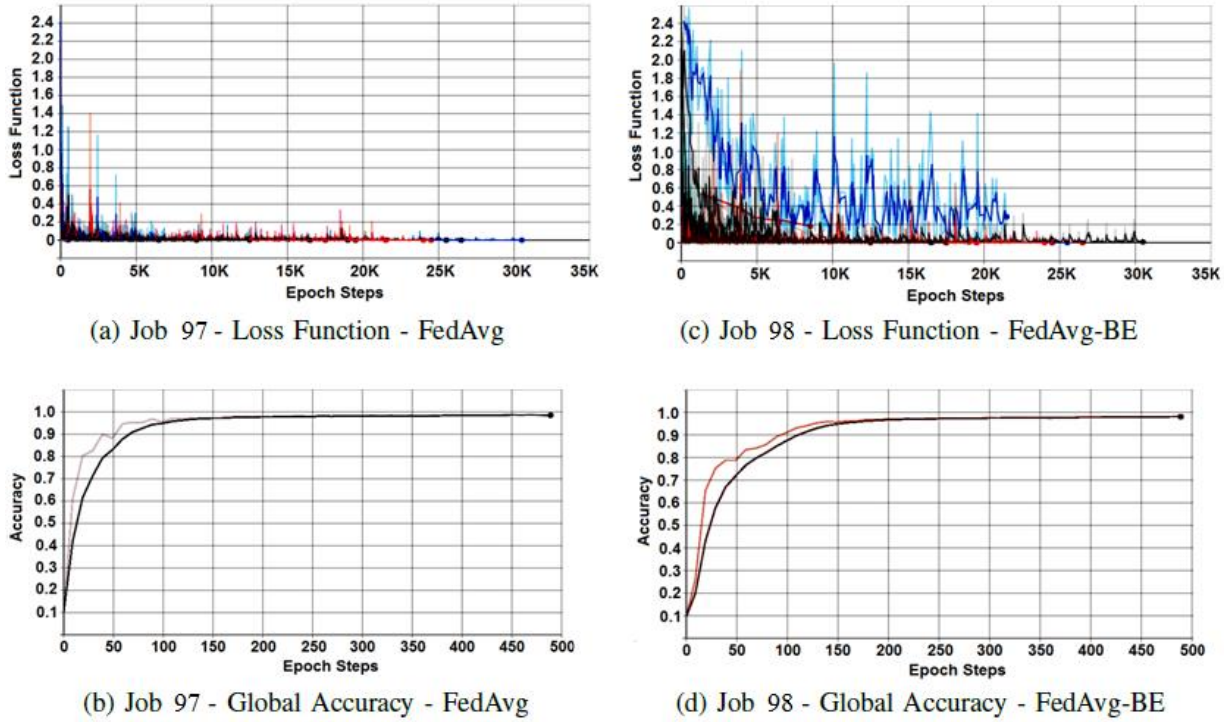
(a) Job 91 - Loss Function - FedAvg

(c) Job 92 - Loss Function - FedAvg-BE

(b) Job 91 - Global Accuracy - FedAvg

(d) Job 92 - Global Accuracy - FedAvg-BE

Figura 34: Gráficos de resultados para o dataset MNIST com SGD em 19 clientes, comparando o modelo FedAvg com o modelo proposto FedAvg-BE



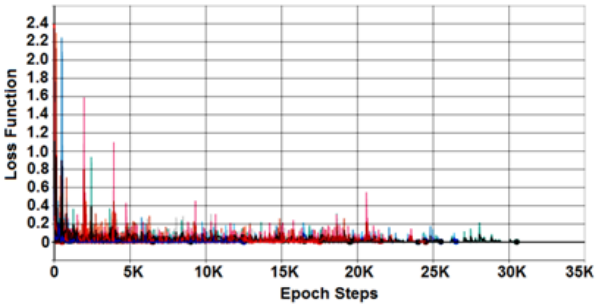
(a) Job 97 - Loss Function - FedAvg

(c) Job 98 - Loss Function - FedAvg-BE

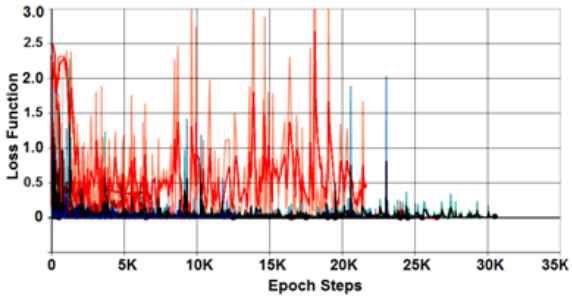
(b) Job 97 - Global Accuracy - FedAvg

(d) Job 98 - Global Accuracy - FedAvg-BE

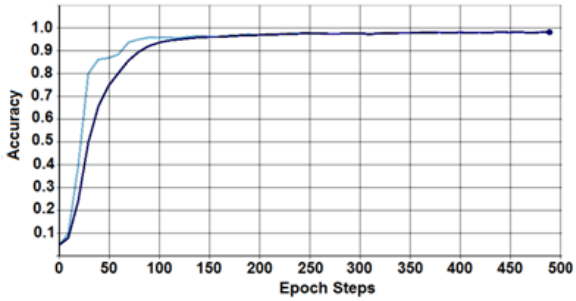
Figura 35: Gráficos de resultados para o dataset MNIST com Adam em 19 clientes, comparando o modelo FedAvg com o modelo proposto FedAvg-BE



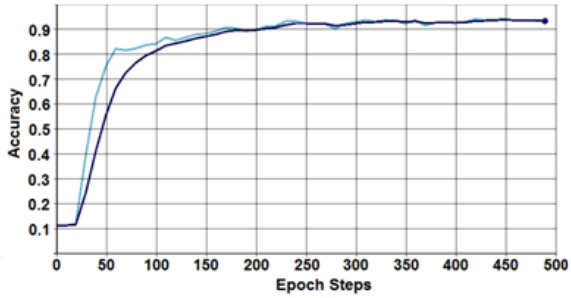
(a) Job 99 - Loss Function - FedAvg



(c) Job 100 - Loss Function - FedAvg-BE

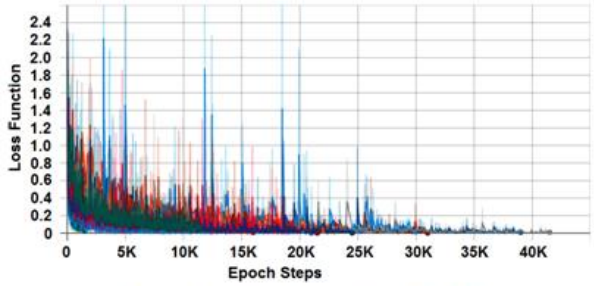


(b) Job 99 - Global Accuracy - FedAvg

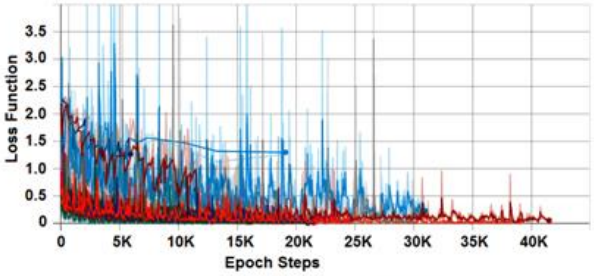


(d) Job 100 - Global Accuracy - FedAvg-BE

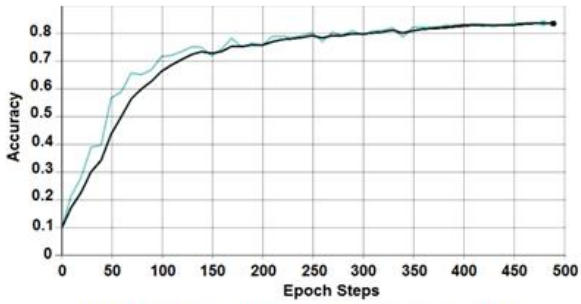
Figura 36: Gráficos de resultados para o dataset MNIST com RMSprop em 19 clientes, comparando o modelo FedAvg com o modelo proposto FedAvg-BE



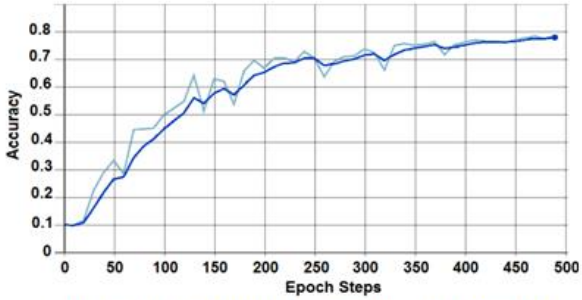
(a) Job 106 - Loss Function - FedAvg



(c) Job 107 - Loss Function - FedAvg-BE

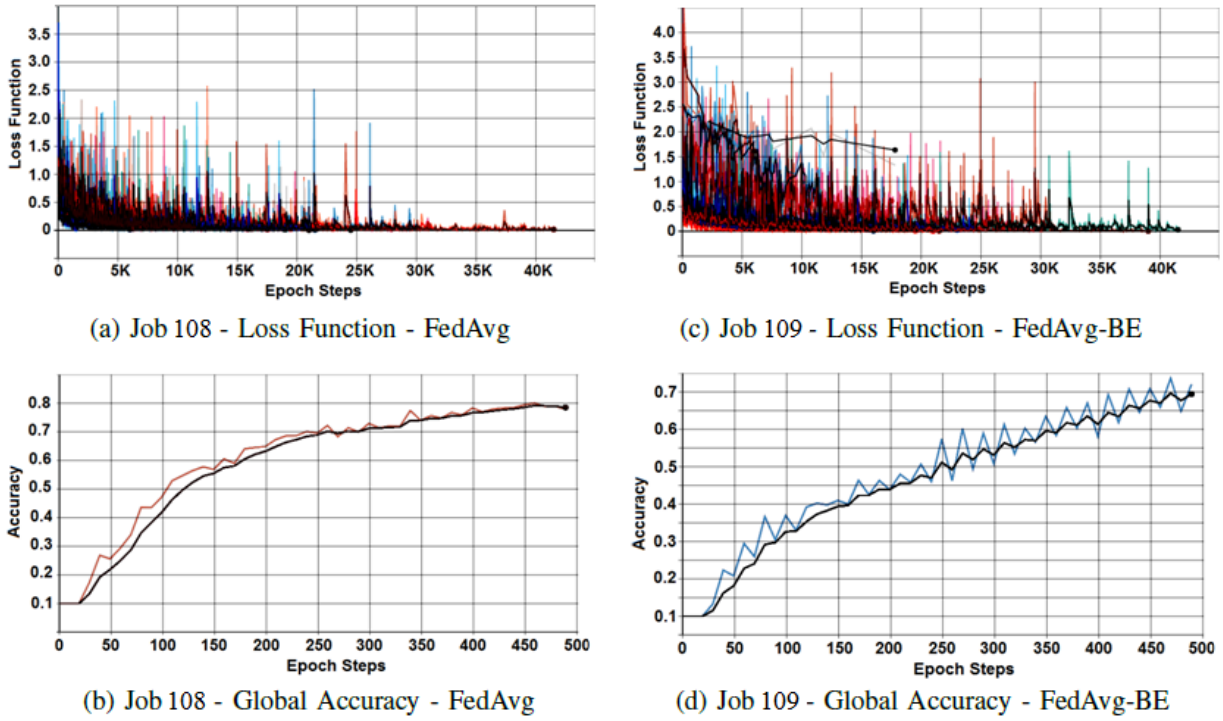


(b) Job 106 - Global Accuracy - FedAvg



(d) Job 107 - Global Accuracy - FedAvg-BE

Figura 37: Gráficos de resultados para o dataset CIFAR-10 com SGD em 19 clientes, comparando o modelo FedAvg com o modelo proposto FedAvg-BE



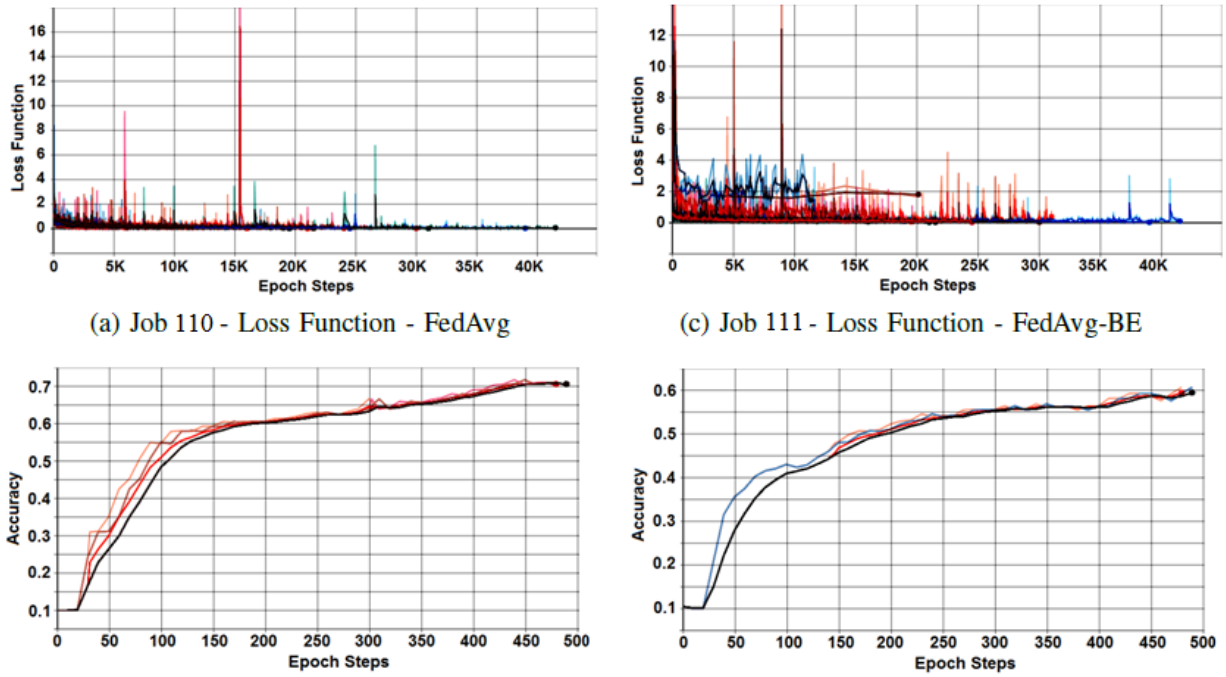
(a) Job 108 - Loss Function - FedAvg

(c) Job 109 - Loss Function - FedAvg-BE

(b) Job 108 - Global Accuracy - FedAvg

(d) Job 109 - Global Accuracy - FedAvg-BE

Figura 38: Gráficos de resultados para o dataset CIFAR-10 com Adam em 19 clientes, comparando o modelo FedAvg com o modelo proposto FedAvg-BE



(a) Job 110 - Loss Function - FedAvg

(c) Job 111 - Loss Function - FedAvg-BE

(b) Job 110 - Global Accuracy - FedAvg

(d) Job 111 - Global Accuracy - FedAvg-BE

Figura 39: Gráficos de resultados para o dataset CIFAR-10 com RMSprop em 19 clientes, comparando o modelo FedAvg com o modelo proposto FedAvg-BE

Para os 19 clientes, a Tabela 13 apresenta a variação dos resultados obtidos com o algoritmo SGD, que mostrou melhor resultado em acurácia, tanto para execução do modelo FedAvg quanto para o modelo proposto FedAvg-BE, incluindo cálculo de entropia. Os *Jobs* 91 e 92 do dataset MNIST e os *Jobs* 106 e 107 do dataset CIFAR-10 são os casos mais significativos obtidos dos experimentos com 19 clientes, considerando a estabilidade da convergência do SGD com a variável *Momentum* em 0,9 e o uso da função *Batch Normalization*. Ao observar esses *Jobs*, nota-se que para o dataset MNIST ocorreu redução do tempo de execução em 5 minutos, que representa redução em torno de 12%, e redução de menos de 1% para a acurácia. Para o dataset CIFAR-10 ocorreu redução do tempo de execução em torno de 30 minutos, que representa redução em torno de 8%, e redução entre 4% e 5% para a acurácia.

Assim como na Tabela 11, onde se apresenta os resultados para 10 clientes, na Tabela 13, que representa resultados para 19 clientes em ambiente de Federated Learning, as notas mostram que houve uma leve diferença na convergência do algoritmo de otimização SGD e na função de perda, devido à seleção de dados, quando aplicado o cálculo de entropia. Com isso, é possível comparar o resultado dos experimentos com o modelo FedAvg e com o modelo FedAvg-BE, para selecionar dados com menor entropia.

Tabela 13: Comparativo da variação dos resultados para os datasets MNIST e CIFAR-10 usando SGD, em 19 clientes

Dataset	**Métrica**	**Resultado FedAvg**	**Resultado FedAvg-BE**	**Notas**
MNIST	Job	91	92	
	Acurácia	98,88%	98,31%	↓ 0,57%
	Tempo de Execução	39 min	34 min	↓ 5 min 12%
CIFAR-10	Job	106	107	
	Acurácia	83,47%	78,68%	↓ 4,79%
	Tempo de Execução	380 min	350 min	↓ 30 min 8%

6.2 Discussão

Para a pesquisa experimental deste trabalho foi necessário implementar aplicações com o SDK NVFlare. Isto possibilitou executar experimentos, no sistema distribuído descrito na

metodologia deste trabalho. Os experimentos serviram para comparar o comportamento dos modelos de Federated Learning e atestar a hipótese de que há relação entre entropia e dados *non*-IID. Em um dos modelos foi implementado cálculo de entropia para treinar dados *non*-IID. Foram avaliadas métricas de software como tempo total de execução, acurácia global, acurácia local por unidade e função de perda. Avaliou-se também que dos 3 tipos de algoritmos de otimização utilizados nos experimentos deste trabalho, o SGD mostrou melhor desempenho em termos de acurácia global e menor função de perda. A maioria dos treinamentos dos modelos FL realizados, neste trabalho, com o algoritmo de otimização SGD tende a ter maior acurácia do que os algoritmos Adam e RMSprop, para os dois conjuntos de dados MNIST e CIFAR-10 e para qualquer quantidade de VMs clientes.

Ao usar cálculo de entropia no modelo de Federated Learning com algoritmo de otimização SGD, menos dados são selecionados e processados, reduzindo levemente a acurácia entre 4% e 5%, independentemente do conjunto de dados e número de clientes. Também foi possível observar redução do tempo de execução do modelo entre 22% a 26%, para 10 clientes e redução do tempo de execução do modelo entre 8% a 12%, para 19 clientes, conforme avaliação das Figuras 27, 28, 29, 30, 31 e 32. Com aumento do número de 10 para 19 clientes e preparação dos dados *non*-IID para os mesmos, houve uma heterogeneidade maior dos dados distribuídos entre os clientes, que não afetou o resultado da acurácia, apenas leve diferença no tempo de execução entre 10 e 19 clientes, mantendo a redução do mesmo. A redução de tempo de execução auxilia para menor consumo de energia em um dispositivo IoT, por exemplo.

Com isso, os resultados com modelo FedAvg adaptado com calculo de entropia, FedAvg-BE, se mostraram aceitáveis e consistentes para a proposta de estudo deste trabalho. Os estudos realizados tanto na revisão sistemática de literatura como os resultados obtidos na pesquisa experimental servem como contribuição para a comunidade científica.

6.3 Considerações Sobre o Capítulo

Neste capítulo foram apresentados os experimentos e seus resultados, descrevendo as divsersas configurações utilizadas para a elaboração dos experimentos. Também foram descritos os detalhes da análise dos resultados, mostrando que os mesmos foram aceitáveis e consistentes para o modelo proposto deste trabalho.

7 CONCLUSÃO

O objetivo deste trabalho foi apresentar um estudo de avaliação de precisão dos modelos disponíveis de Federated Learning e implementação do modelo FedAvg-BE, para validar a hipótese de uma relação direta entre entropia, ou seja, o grau de incerteza, e dados *non*-IID. Foram realizados experimentos, usando SDK NVFlare e o framework PyTorch, com dois conjuntos de dados mais utilizados no meio acadêmico para testes de Machine Learning, distribuindo os mesmos em 19 unidades clientes e 1 unidade servidora. Em cada uma das unidades foram executados experimentos, variando conjunto de dados, rede neural CNN, algoritmos de otimização e cálculo de entropia.

Os experimentos mediram o impacto do cálculo de entropia, assim foram contempladas algumas métricas de software, como tempo total de execução, acurácia global, acurácia local por unidade e função de perda. Mostrou-se que a maioria dos treinamentos de modelo realizados com o algoritmo de otimização SGD tende a ser mais preciso do que os algoritmos Adam e RMSprop, para os dois conjuntos de dados MNIST e CIFAR-10. Ao usar cálculo de entropia, menos dados são selecionados e processados, reduzindo levemente a acurácia entre 4% e 5% como também reduz o tempo de execução do modelo, independente do conjunto de dados.

Por fim, é seguro dizer que o modelo FedAvg-BE, que usa a entropia, junto com algoritmo de otimização SGD, se apresenta como melhor escolha para reduzir o tempo de execução de aplicações de Federated Learning e auxilia na redução do consumo de energia para dispositivos IoT. Os resultados dos experimentos se mostraram aceitáveis e consistentes para a proposta de estudo deste trabalho.

7.1 Contribuições

Este trabalho contribuiu elaborando uma revisão sistemática de literatura, pesquisando modelos existentes de Federated Learning, aplicados em ambiente de sistemas autônomos distribuídos com dados heterogêneos. Com os objetivos, questões, estratégias de pesquisa da RSL, foram selecionadas 30 publicações mais relevantes, onde foi possível identificar modelos de Federated Learning, tipos de algoritmo para Big Data e dados *non*-IID, tecnologias para desenvolver sistemas para dispositivos IoT e métricas utilizadas. Em alguns trabalhos relacionados recentes, o foco foi sobre o aspecto de entropia dos dados. Um dos trabalhos criou modelo com base no julgamento de máxima entropia, outro trabalho criou modelo com aprendizado semi-supervisionado que aprimora as saídas com média de redução de entropia

para tratar dados *non*-IID e reduzir custo de comunicação. Neste trabalho o modelo proposto FedAvg-BE procurou tratar os dados *non*-IID com entropia e reduzir tempo de execução.

As principais contribuições realizadas neste trabalho quanto a resposta à questão de pesquisa e os objetivos estabelecidos, são listadas a seguir:

- Avaliação da influência da entropia sobre dados *non*-IID em ambiente de Federated Learning;
- Elaboração de um mecanismo consistente, sendo independente do tipo de dataset para ajustar a entrada de dados ao processamento eficiente de dispositivos IoT;
- Elaboração de uma metodologia de avaliação e seleção de dados, sem causar grande impacto na comunicação existente entre os dispositivos no ambiente federado;
- Desenvolvimento de um modelo de solução que permite a criação de aplicações Federated Learning que avaliam a qualidade dos dados para ser processado somente o que agregar maior valor à informação;
- Criação de um algoritmo para o tratamento da informação que extenda o modelo de aplicações Federated Learning mitigando os efeitos de dados *non*-IID.

Como contribuição adicional também se destaca a avaliação experimental deste trabalho com modelos de Federated Learning para atestar a hipótese de que a entropia tem relação direta com dados *non*-IID e fornecer uma comparação entre modelos de Federated Learning e algoritmos de otimização, implementando cálculo de entropia em um dos modelos e usando dados do tipo *non*-IID, ou seja, dados heterogêneos. Outra contribuição são as definições de configuração da arquitetura do mecanismo de rede para auxiliar os futuros usuários do SDK NVFlare da NVIDIA na instalação e configuração dos ambientes em máquinas virtuais, auxiliando quem deseja obter os códigos para pesquisas experimentais com modelos de Federated Learning.

7.2 Trabalhos Futuros

Embora os resultados apresentados tenham cumprido os objetivos previamente estabelecidos e respondido as questões de pesquisa, percebe-se a oportunidade de realizar melhorias adicionais na pesquisa experimental realizada. Com relação a trabalhos futuros, vale destacar:

- Realizar mais experimentos, verificando uma forma de ajustar o modelo FedAvg-BE para otimizar a acurácia com cálculo de entropia, talvez combinando outros modelos de Federated Learning, como por exemplo FedOpt.

- A introdução de pelo menos um modelo de unidade RNN ou LSTM no âmbito das experiências futuras.
- Adição de mais conjuntos de dados conhecidos ou um conjunto de dados personalizado.
- Instalar e configurar as estruturas com suporte de processamento para treinamento distribuído de várias GPUs.
- Instalar e configurar alguma ferramenta que funcione em conjunto com SDK NVFlare para avaliar consumo de energia nos experimentos.
- Analisar um número variável de dispositivos clientes, gerando gráficos comparativos com número de clientes.
- Avaliar e testar índice de Gini como alternativa para a entropia, na determinação da heterogeneidade ou homogeneidade dos rótulos de uma classe.

REFERÊNCIAS

ABDULRAHMAN, Sawsan; TOUT, Hanine; OULD-SLIMANE, Hakima; MOURAD, Azzam; TALHI, Chamseddine; GUIZANI, Mohsen. (2020). **A Survey on Federated Learning: The Journey From Centralized to Distributed On-Site Learning and Beyond**. IEEE Internet of Things Journal. PP. 10.1109/JIOT.2020.3030072.

ALLENDE-CID, Héctor. **Machine Learning: Catalisador da Ciência**, in Computação Brasil, Revista da Sociedade Brasileira de Computação, v. 39, Ed. 01, 2019.

ALZUBAIDI, L., ZHANG, J., HUMAIDI, A.J. et al. **Review of deep learning: concepts, CNN architectures, challenges, applications, future directions**. J Big Data 8, 53 (2021). https://doi.org/10.1186/s40537-021-00444-8

BOTTOU, Léon; CURTIS, Frank E.; NOCEDAL, Jorge. **Optimization Methods for Large-Scale Machine Learning**. arXiv:1606.04838v3. 2018.

CATAL, Cagatay. **Performance Evaluation Metrics for Software Fault Prediction Studies.** Acta Polytechnica Hungarica. Vol. 9, No. 4, 2012.

CERRI, R.; CARVALHO, A. C. P. de L. F. **Aprendizado de Máquina: Breve Introdução e Aplicações**, in Cadernos de Ciência & Tecnologia, Brasília, v. 34, 2017.

CHAGAS, Edgar Thiago De Oliveira. **Deep Learning e suas aplicações na atualidade**. Revista Científica Multidisciplinar Núcleo do Conhecimento. Ano 04, Ed. 05, Vol. 04, pp. 05-26 Maio de 2019. ISSN: 2448-0959.

CHEN, Z.; TIAN, P.; LIAO, W.; YU, W. **Zero Knowledge Clustering Based Adversarial Mitigation in Heterogeneous Federated Learning**, in IEEE Transactions on Network Science and Engineering, doi: 10.1109/TNSE.2020.3002796.

CHIU, Te-Chuan; SHIH, Yuan-Yao; PANG, Ai-Chun; WANG, Chieh-Sheng; WENG, Wei; CHOU, Chun-Ting. (2020). **Semisupervised Distributed Learning with Non-IID Data for AIoT Service Platform**. IEEE Internet of Things Journal. PP. 1-1. 10.1109/JIOT.2020.2995162.

CLAUSET, Aaron. **Inference, Models and Simulation for Complex Systems** - Lecture 0 (PDF). 2011. Santa Fe Institute. https://aaronclauset.github.io/courses/7000/csci7000-001_2011_L0.pdf . Acessado em 02 de março de 2023.

COSTA, S.; SANTOS, S. A.; STRAPASSON, J. **Fisher information distance: a geometrical reading**, ArXiv, abs/1210.2354 (2014): n. pag.

DENNIS, L. A.; FISHER, M. **Verifiable Self-Aware Agent-Based Autonomous Systems**, in Proceedings of the IEEE, vol. 108, no. 7, pp. 1011-1026, July 2020, doi: 10.1109/JPROC.2020.2991262.

DU, Z.; WU, C.; YOSHINAGA, T.; YAU, K. A.; JI, Y.; LI, J. **Federated Learning for Vehicular Internet of Things: Recent Advances and Open Issues**, in IEEE Open Journal of the Computer Society, vol. 1, pp. 45-61, 2020, doi: 10.1109/OJCS.2020.2992630.

EDUREKA BLOG. https://www.edureka.co/blog/data-science-vs-machine-learning . 2020. Acessado em 14/03/2023.

GAO, X.; WANG, D.; ZHANG, J.; LIAO, Q.; LIU, B. **iRBP-Motif-PSSM: Identification of RNA-Binding Proteins Based on Collaborative Learning**, in IEEE Access, vol. 7, pp. 168956-168962, 2019, doi: 10.1109/ACCESS.2019.2952621.

GHOLAMY, A.; KREINOVICH, V.; KOSHELEVA, O. (2018). **Why 70/30 or 80/20 Relation Between Training and Testing Sets: A Pedagogical Explanation**.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning.** MIT Press. 2017.

GOGGLE BLOG. https://ai.googleblog.com/2017/04/federated-learning-collaborative.html. 2017. Acessado em 23/07/2023.

HELLSTRÖM, Henrik & Barros da Silva Júnior, José Mairton & Fodor, Viktoria & Fischione, Carlo. **Wireless for Machine Learning**. 2020. Disponível em: https://doi.org/10.48550/arxiv.2008.13492 . Acessado em 02/03/2023.

HU, Shuyan; CHEN, Xiaojing; NI, Wei; HOSSAIN, Ekram; WANG, Xin. (2021). **Distributed Machine Learning for Wireless Communication Networks: Techniques, Architectures, and Applications**. IEEE Communications Surveys & Tutorials. PP. 1-1. 10.1109/COMST.2021.3086014.

IMANI, M. *et al*. **A Framework for Collaborative Learning in Secure High-Dimensional Space**, 2019 IEEE 12th International Conference on Cloud Computing (CLOUD), Milan, Italy, 2019, pp. 435-446, doi: 10.1109/CLOUD.2019.00076.

ITAHARA, S. et al. **Distillation-Based Semi-Supervised Federated Learning for Communication-Efficient Collaborative Training With Non-IID Private Data**. In IEEE Transactions on Mobile Computing, vol. 22, no. 01, pp. 191-205, 2023, doi: 10.1109/TMC.2021.3070013.

JIANG, Jiawei and CUI, Bin and ZHANG, Ce. **Distributed Machine Learning and Gradient Optimization**. Big Data Management, 2022, doi: 10.1007/978-981-16-3420-8.

JUNIOR, Jonas Bach. **O pensar intuitivo como fundamento de uma educação para a liberdade**. In Educar em Revista, Curitiba, Brasil, n. 56, p. 131-145, abr./jun. 2015. Editora UFPR. DOI: 10.1590/0104-4060.41520.

KARIMIREDDY, S. P.; KALE, S.; MOHRI, M.; REDDI, S. J.; STICH, S. U.; SURESH, A. T. **SCAFFOLD: Stochastic Controlled Averaging for Federated Learning**. In arXiv, 2021, doi; 10.48550/ARXIV.1910.06378.

KIM, J. et al. **A Novel Joint Dataset and Computation Management Scheme for Energy-Efficient Federated Learning in Mobile Edge Computing**. In IEEE Wireless Communications Letters, vol. 11, no. 5, pp. 898-902, May 2022, doi: 10.1109/LWC.2022.3147236.

KITCHENHAM, Barbara. **Procedures for Performing Systematic Reviews**. In: (2004).

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. In: 2 (Jan. 2007).

KHOA, T. V. *et al*. **Collaborative Learning Model for Cyberattack Detection Systems in IoT Industry 4.0**, 2020 IEEE Wireless Communications and Networking Conference (WCNC), Seoul, Korea (South), 2020, pp. 1-6, doi: 10.1109/WCNC45663.2020.9120761.

LEI, L.; TAN, Y.; ZHENG, K.; LIU, S.; ZHANG, K.; SHEN, X. **Deep Reinforcement Learning for Autonomous Internet of Things: Model, Applications and Challenges**, in IEEE Communications Surveys & Tutorials, vol. 22, no. 3, pp. 1722-1760, thirdquarter 2020, doi: 10.1109/COMST.2020.2988367.

LI, B.; JIANG, Y.; PEI, Q.; LI, T.; LIU, L; LU, R. **FEEL: Federated End-to-End Learning With Non-IID Data for Vehicular Ad Hoc Networks**. In IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 9, pp. 16728-16740, Sept. 2022, doi: 10.1109/TITS.2022.3190294.

LI, B.; MA, S.; DENG, R.; CHOO, K. -K. R; YANG, J. **Federated Anomaly Detection on System Logs for the Internet of Things: A Customizable and Communication-Efficient Approach**. In IEEE Transactions on Network and Service Management, vol. 19, no. 2, pp. 1705-1716, June 2022, doi: 10.1109/TNSM.2022.3152620.

LI, T.; SAHU, A. K.; TALWALKAR, A.; SMITH, V. **Federated Learning: Challenges, Methods, and Future Directions**. IEEE Signal Processing Magazine, vol. 37, no. 3, p. 50–60, May 2020.

LING, Z. et al. **FedEntropy: Efficient Device Grouping for Federated Learning Using Maximum Entropy Judgment**. ArXiv abs/2205.12038 (2022): n. pag.

MACHADO, Felipe. **Big Data O Futuro dos Dados e Aplicações**. 2018. São Paulo: Érica. pp. 44–46

McMAHAN, H.B.; MOORE, E.; RAMAGE, D.; HAMPSON, S.; ARCAS, B.A. (2017). **Communication-Efficient Learning of Deep Networks from Decentralized Data**. *AISTATS*.

MENG, L.; YOU, X.; LIU, S. **Multi-Colony Collaborative Ant Optimization Algorithm Based on Cooperative Game Mechanism**, in IEEE Access, vol. 8, pp. 154153-154165, 2020, doi: 10.1109/ACCESS.2020.3011936.

MESCHEDER, L.; OECHSLE, M.; NIEMEYER, M.; NOWOZIN, S.; GEIGER, A. **Occupancy Networks: Learning 3D Reconstruction in Function Space**. In: 2019, arXiv: 1812.03828.

NORDLUND, N.; KWON, H.; TASSIULAS, L. **Cooperative Learning for Multi-perspective Image Classification**, 2019 IEEE International Conference on Smart Computing (SMARTCOMP), Washington, DC, USA, 2019, pp. 75-80, doi: 10.1109/SMARTCOMP.2019.00032.

NVFLARE. https://nvflare.readthedocs.io/en/2.0/index.html. 2022. Acessado em 23/01/2023.

ORLANDI, F. C.; ANJOS, J. C. S.; LEITHARDT, V. R. Q.; SANTANA, J. F. de P.; GEYER, C. F. R. **Entropy to mitigate non-IID data problem on Federated Learning for the Edge Intelligence environment**. In IEEE Access, 2023, doi: 10.1109/ACCESS.2023.3298704.

PATGIRI, Ripon. **Taxonomy of Big Data: A Survey**. 2019. arXiv:1808.08474v3.

PAVIOTTI, José Renato. **Considerações sobre o conceito de entropia na teoria da informação**. Unicamp – Limeira, SP. 2019.

PENG, Y.; ZHAO, Y.; ZHANG, J. **Two-Stream Collaborative Learning With Spatial-Temporal Attention for Video Classification**, in IEEE Transactions on Circuits and Systems for Video Technology, vol. 29, no. 3, pp. 773-786, March 2019, doi: 10.1109/TCSVT.2018.2808685.

PYTORCH. https://pytorch.org/docs/stable/index.html. 2022. Acessado em 23/01/2023.

QIAO, D. et al. **Adaptive Federated Deep Reinforcement Learning for Proactive Content Caching in Edge Computing**. In IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 12, pp. 4767-4782, 1 Dec. 2022, doi: 10.1109/TPDS.2022.3201983.

RAO, A. S.; GEORGEFF, M. P. **BDI-agents: from theory to practice**, in Proceedings of the First Intl. Conference on Multiagent Systems, 1995, San Francisco.

REDDI, S.; CHARLES, Z.; ZAHEER, M.; GARRETT, Z.; RUSH, K.; KONEČNÝ, J.; KUMAR, S.; MCMAHAN, H. B. **Adaptive Federated Optimization**. In arXiv, 2021, doi: 10.48550/ARXIV.2003.00295.

RUBY, R. et al. **Energy-Efficient Multiprocessor-Based Computation and Communication Resource Allocation in Two-Tier Federated Learning Networks**. In IEEE Internet of Things Journal, vol. 10, no. 7, pp. 5689-5703, 1 April1, 2023, doi: 10.1109/JIOT.2022.3153996.

SAHU, A.; LI, T.; SANJABI, M.; ZAHEER, M.; TALWALKAR, A.S.; SMITH, V. (2020). **Federated Optimization in Heterogeneous Networks**. arXiv: Learning.

SEO, J.; PARK, H. **Object Recognition in Very Low Resolution Images Using Deep Collaborative Learning**, in IEEE Access, vol. 7, pp. 134071-134082, 2019, doi: 10.1109/ACCESS.2019.2941005.

SHAO, T.; Kui, X.; ZHANG, P.; CHEN, H. **Collaborative Learning for Answer Selection in Question Answering**, in IEEE Access, vol. 7, pp. 7337-7347, 2019, doi: 10.1109/ACCESS.2018.2890102.

SHEENA; KUMAR, Krishan, & KUMAR, Gulshan. **Analysis of Feature Selection Techniques: A Data Mining Approach**. 2016.

VERBRAEKEN, Joost; WOLTING, Matthijs; KATZY, Jonathan; KLOPPENBURG, Jeroen; VERBELEN, Tim; and RELLERMEYER, Jan S. 2020. **A Survey on Distributed Machine Learning**. ACM Comput. Surv. 53, 2, Article 30 (March 2021), 33 pages. DOI:https://doi.org/10.1145/3377454

WANG, D.; CHEN, I.; AL-HAMADI, H. **Reliability of Autonomous Internet of Things Systems With Intrusion Detection Attack-Defense Game Design**, in IEEE Transactions on Reliability, doi: 10.1109/TR.2020.2983610.

WANG, Hao; KAPLAN, Zakhary; NIU, Di; LI, Baochun. (2020). **Optimizing Federated Learning on Non-IID Data with Reinforcement Learning**. 1698-1707. 10.1109/INFOCOM41043.2020.9155494.

WEI, X.; ZHAO, J.; ZHOU, L.; QIAN, Y. **Broad Reinforcement Learning for Supporting Fast Autonomous IoT**, in IEEE Internet of Things Journal, vol. 7, no. 8, pp. 7010-7020, Aug. 2020, doi: 10.1109/JIOT.2020.2980198.

WU, C.; LIU, Z.; LIU, F.; YOSHINAGA, T.; JI, Y.; LI, J. **Collaborative Learning of Communication Routes in Edge-enabled Multi-access Vehicular Environment**, in IEEE Transactions on Cognitive Communications and Networking, doi: 10.1109/TCCN.2020.3002253.

WU, Q.; CHEN, X.; ZHOU, Z.; ZHANG, J. **FedHome: Cloud-Edge Based Personalized Federated Learning for In-Home Health Monitoring**. In IEEE Transactions on Mobile Computing, vol. 21, no. 8, pp. 2818-2832, 1 Aug. 2022, doi: 10.1109/TMC.2020.3045266.

XIE, Y.; XIA, Y.; ZHANG, J.; SONG, Y.; FENG, D.; FULHAM, M.; CAI, W. **Knowledge-based Collaborative Deep Learning for Benign-Malignant Lung Nodule Classification on Chest CT**. IEEE Trans Med Imaging. 2019 Apr;38(4):991-1004. doi: 10.1109/TMI.2018.2876510. Epub 2018 Oct 17. PMID: 30334786.

YOU, X.; LIU, X.; JIANG, N.; CAI, J.; YING, Z. **Reschedule Gradients: Temporal Non-IID Resilient Federated Learning**. In IEEE Internet of Things Journal, vol. 10, no. 1, pp. 747-762, 1 Jan.1, 2023, doi: 10.1109/JIOT.2022.3203233.

ZHANG, J. *et al*. **Adaptive Federated Learning on Non-IID Data With Resource Constraint**. In IEEE Transactions on Computers, vol. 71, no. 7, pp. 1655-1667, 1 July 2022, doi: 10.1109/TC.2021.3099723.

ZHANG, Wenyu; WANG, Xiumin; ZHOU, Pan; WU, Weiwei; ZHANG, Xinglin. (2021). **Client Selection for Federated Learning With Non-IID Data in Mobile Edge Computing**. IEEE Access. PP. 1-1. 10.1109/ACCESS.2021.3056919.

ZHANG, Y. et al. **FedMDS: An Efficient Model Discrepancy-Aware Semi-Asynchronous Clustered Federated Learning Framework**. In IEEE Transactions on Parallel and Distributed Systems, vol. 34, no. 3, pp. 1007-1019, 1 March 2023, doi: 10.1109/TPDS.2023.3237752.

ZHANG, Z. et al. **GOF-TTE: Generative Online Federated Learning Framework for Travel Time Estimation**. In IEEE Internet of Things Journal, vol. 9, no. 23, pp. 24107-24121, 1 Dec.1, 2022, doi: 10.1109/JIOT.2022.3190864.

ZHAO, L.; WANG, Q.; ZOU, Q.; ZHANG, Y.; CHEN, Y. **Privacy-Preserving Collaborative Deep Learning With Unreliable Participants**, in IEEE Transactions on Information Forensics and Security, vol. 15, pp. 1486-1500, 2020, doi: 10.1109/TIFS.2019.2939713.

ZHAO, Z. et al. **Federated Learning With Non-IID Data in Wireless Networks**. In IEEE Transactions on Wireless Communications, vol. 21, no. 3, pp. 1927-1942, March 2022, doi: 10.1109/TWC.2021.3108197.

ZHU, Hangyu; XU, Jinjin; LIU, Shiqing; JIN, Yaochu. (2021). **Federated Learning on Non-IID Data: A Survey.** https://doi.org/10.1016/j.neucom.2021.07.098.