UNIVERSIDADE FEDERAL DE GOIÁS / INSTITUTO DE INFORMÁTICA

Aprendizado Federado em Ambiente Distribuído

Avaliação de Algoritmos para Dados Heterogêneos

Marcelo Henrique Alves Pereira Sobrinho

UNIVERSIDADE FEDERAL DE GOIÁS (UFG)
INSTITUTO DE INFORMÁTICA (INF)

MARCELO HENRIQUE ALVES PEREIRA SOBRINHO

Aprendizado Federado em Ambiente Distribuído

Avaliação de Algoritmos para Dados Heterogêneos

Goiânia
2025

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO NO REPOSITÓRIO INSTITUCIONAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio do Repositório Institucional (RI/UFG), regulamentado pela Resolução CEPEC no 1240/2014, sem ressarcimento dos direitos autorais, de acordo com a Lei no 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo dos Trabalhos de Conclusão dos Cursos de Graduação disponibilizado no RI/UFG é de responsabilidade exclusiva dos autores. Ao encaminhar(em) o produto final, o(s) autor(a)(es)(as) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do Trabalho de Conclusão de Curso de Graduação (TCCG)

Nome(s) completo(s) do(a)(s) autor(a)(es)(as): MARCELO HENRIQUE ALVES PEREIRA SOBRINHO

Título do trabalho: Aprendizado Federado em Ambiente Distribuído

Avaliação de Algoritmos para Dados Heterogêneos

2. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador) Concorda com a liberação total do documento [X] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante: a) consulta ao(à)(s) autor(a)(es)(as) e ao(à) orientador(a); b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo do TCCG. O documento não será disponibilizado durante o período de embargo.

Casos de embargo:

- Solicitação de registro de patente;

- Submissão de artigo em revista científica;

- Publicação como capítulo de livro.

Obs.: Este termo deve ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Marcelo Henrique Alves Pereira Sobrinho**, **Discente**, em 15/04/2026, às 17:02, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Fernando Marques Federson**, **Professor do Magistério Superior**, em 15/04/2026, às 18:53, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **5956756** e o código CRC **D03181C7**.

Referência: Processo nº 23070.005522/2026-17 SEI nº 5956756

MARCELO HENRIQUE ALVES PEREIRA SOBRINHO

Aprendizado Federado em Ambiente Distribuído

Avaliação de Algoritmos para Dados Heterogêneos

Relatório final de Trabalho de Conclusão de Curso, apresentado à Universidade Federal de Goiás, como parte das exigências para a obtenção do título de Bacharel em Inteligência Artificial.

Orientador: Prof. Dr. Fernando Marques Federson

Goiânia

2025

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UFG.

SOBRINHO, MARCELO HENRIQUE ALVES PEREIRA
Aprendizado Federado em Ambiente Distribuído [manuscrito]: Avaliação de Algoritmos para Dados Heterogêneos / MARCELO HENRIQUE ALVES PEREIRA SOBRINHO. - 2025.
83 f.: 2025

Orientador: Prof. Dr. Fernando Marques Federson
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Goiás, Instituto de Informática (INF), Inteligência Artificial, Goiânia, 2025.

1. Inteligência Artificial. 2. Aprendizado Federado. 3. Dados Heterogêneos.

I. Federson, Fernando Marques , orient. II. Título.

CDU 004

MARCELO HENRIQUE ALVES PEREIRA SOBRINHO

Aprendizado Federado em Ambiente Distribuído

Avaliação de Algoritmos para Dados Heterogêneos

Relatório final de Trabalho de Conclusão de Curso, apresentado à Universidade Federal de Goiás, como parte das exigências para a obtenção do título de Bacharel em Inteligência Artificial.

Data da Aprovação: 09 de dezembro de 2025.

Prof. Dr. Fernando Marques Federson
Orientador (INF-UFG)

Prof. Dr. Aldo André Díaz Salazar
Coordenador de TCC do BIA (INF-UFG)

Prof. Dr. Anderson da Silva Soares
Coordenador do BIA (INF-UFG)

Prof. Dr. Sávio Salvarino Teles de Oliveira
(INF-UFG)

MARCELO HENRIQUE ALVES PEREIRA SOBRINHO

Aprendizado Federado em Ambiente Distribuído

Avaliação de Algoritmos para Dados Heterogêneos

RESUMO

Este Relatório de Conclusão de Curso tem como objetivo reunir os resultados da minha jornada para me tornar um especialista em **Aprendizado Federado**. Uma ilustração e sua narrativa descrevem os períodos de trabalho. Os Apêndices contêm os Termos de Aceite de Entrega e os resultados obtidos durante cada período de trabalho.

Palavras-chave: Inteligência artificial; Aprendizado federado; Dados heterogêneos.

ABSTRACT

This Course Completion Report aims to bring together the results of my journey to become an expert in **Federated Learning**. An illustration and its narrative describe the work periods. The Appendices contain the Delivery Acceptance Terms and the results obtained during each work period.

Keywords: Artificial intelligence; Federated learning; Heterogeneous data.

Goiânia

2025

Minha Jornada

Marcelo Henrique Alves

Especialista em: Aprendizado Federado

MINHA JORNADA

Nome: **Marcelo Henrique Alves Pereira Sobrinho**

Especialidade: **Aprendizado Federado**

Objetivo deste documento

Durante o processo da disciplina Residência em IA[1], foram gerados diversos resultados na construção da minha especialização. A cada semana, um conjunto de resultados foi formalizado por um Termo de Aceite de Entrega e avaliado por uma banca, considerando o planejado e o realizado para o período. Este documento tem como objetivo descrever esses resultados obtidos, fazendo referência aos Termos de Aceite de Entrega e seus documentos associados.

Minha Jornada

Minha Jornada teve início nas **Semanas 1 e 2,** onde o objetivo foi explorar a intersecção entre as áreas de Processamento Massivo de Dados, Machine Learning e Computação distribuída. Nesse período, explorei artigos fundamentais da área, como *MapReduce: Simplified Data Processing on Large Clusters* e *The Hadoop Distributed File System***.** A intenção inicial era compreender melhor os princípios de Sistemas Distribuídos para, em seguida, avançar em tópicos mais específicos relacionados à união desses dois temas com à Inteligência Artificial. Foi nesse caminho que encontrei o *Survey on Dsitributed Machine Learning*, cuja seção sobre Privacidade de Dados apresentou o conceito de Aprendizado Federado, no qual eu decidi escolher como tema principal da Residência. Essa trajetória de pesquisa, com referências detalhadas e materiais consultados, pode ser encontrada de forma completa no **Apêndice 1**.

Nas **Semanas 3 e 4**, concentrei meus esforços na consolidação do tema **Aprendizado Federado**, aprofundando a leitura dos principais artigos da área e registrando anotações e

[1] Dez Semanas, entre setembro de 2025 e dezembro de 2025.

resumos que orientaram minha compreensão inicial sobre o assunto. Estudei tanto o artigo seminal *Communication-Efficient Learning of Deep Networks from Decentralized Data*, quanto a revisão *Federated Learning: Challenges, Methods, and Future Directions***,** além de investigar frameworks como NVIDIA FLARE e TensorFlow Federated para experimentações futuras. Também explorei a problemática de dados *não-IID*, identificando o estudo *Federated Learning on Non-IID Data Silos: An Experimental Study* como uma referência essencial para orientar as próximas etapas. As análises detalhadas, incluindo as anotações produzidas durante a leitura dos artigos, estão organizadas de forma completa no **Apêndice 2**.

Nas **Semanas 5 e 6**, aprofundei meu estudo sobre Aprendizado Federado, com foco nos desafios causados por dados *não-IID*. Li e resumi o artigo *Federated Learning on Non IID Data Silos: An Experimental Study*, que compara algoritmos como FedAvg, FedProx, FedNova e SCAFFOLD em diferentes cenários de heterogeneidade e desbalanceamento. Após essa leitura, escolhi esse artigo como base para realizar uma replicação parcial dos experimentos. Paralelamente, avaliei os frameworks TensorFlow Federated e NVIDIA FLARE e optei pelo FLARE devido ao suporte ao PyTorch. Após estudar sua documentação, configurei e testei o ambiente simulado utilizando o CIFAR 10, validando a execução de experimentos básicos. Também pesquisei trabalhos mais recentes, como o artigo sobre o *FedHybrid* de 2025, e revisei conceitos matemáticos importantes de probabilidade e estatística para reforçar os fundamentos teóricos necessários. Todo o material detalhado referente a esta etapa está organizado no **Apêndice 3.**

Nas **Semanas 7 e 8**, dei continuidade ao estudo aprofundado de Aprendizado Federado, com foco especial nos desafios impostos pelos dados *não-IID*. Depois de revisar artigos fundamentais sobre o tema, avancei para a parte prática da replicação do artigo mencionado utilizando o NVIDIA FLARE. Percebi que o framework não realizava o particionamento dos dados automaticamente, então desenvolvi meu próprio método de divisão *não-IID* do CIFAR-10 usando a distribuição de Dirichlet, armazenando as partições em cache para garantir reprodutibilidade. Integrei esse particionamento a um conjunto de scripts customizados para cliente e servidor. Paralelamente, estudei o artigo sobre o *FedHybrid*, que combina vantagens de FedAvg, FedProx e SCAFFOLD, e avancei na

preparação para replicar o estudo *Federated Learning on Non-IID Data Silos*. Para isso, conferi detalhadamente os *datasets* usados no artigo, ajustei o repositório, organizei os modelos necessários e sistematizei os experimentos no documento de resumo. Com isso, finalizei esta etapa com toda a base técnica e estrutural pronta para iniciar a replicação completa dos experimentos. Os resultados estão descritos no **Apêndice 4.**

Na **Semana 9** da Residência em IA, avancei para a fase experimental do estudo de Aprendizado Federado, consolidando todo o trabalho desenvolvido nas **Semanas** anteriores. Depois de estudar Computação Distribuída, compreender os fundamentos do FedAvg no artigo seminal e investigar profundamente o problema dos dados não independentes e não identicamente distribuídos, escolhi o artigo *Federated Learning on Non-IID Data Silos: An Experimental Study* para realizar uma replicação parcial dos experimentos, como já mencionei. Utilizando o framework NVIDIA FLARE, que venho explorando e adaptando ao longo do projeto, conduzi meu primeiro experimento envolvendo cenários em que os usuários possuem quantidades diferentes de dados, analisando, portanto, como essa variação afeta o desempenho dos algoritmos federados. Os resultados qualitativos mostraram comportamentos coerentes com o que foi reportado no artigo, mas também apresentaram discrepâncias esperadas devido às diferenças entre implementações, ambientes e parâmetros utilizados. Além disso, configurei o monitoramento completo do treinamento no *Weights and Biases*, o que facilitou a visualização e análise das curvas de aprendizado ao longo das rodadas de comunicação. Também iniciei a preparação do ambiente para rodar o modelo de forma consistente em múltiplas máquinas em um ambiente distribuído de fato. Os resultados do experimento estão no **Apêndice 5.**

Na **Semana 10**, incluí as métricas de treinamento no documento do experimento. Também iniciei a aplicação distribuída do Aprendizado Federado, percebendo que o modo de simulação tratava mais de paralelismo do que de distribuição real. Estudei os modos de desenvolvimento e produção do NVIDIA-FLARE com apoio quase exclusivo da documentação oficial, já que havia pouca orientação disponível fora da documentação. Consegui realizar uma rodada completa de comunicação conforme a teoria do Aprendizado Federado, configurando máquinas, comunicação, provisionamento, distribuição dos Startup

Kits e envio de um job para o cliente. Todo o processo foi registrado em formato de tutorial e está no **Apêndice 6.**

APÊNDICE 1

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: **4 de set. de 2025**

Participantes da Entrega **[matriculados em Residência em IA]**:

MARCELO HENRIQUE ALVES PEREIRA SOBRINHO

Entrega: **[descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]**

Nessa primeira Semana da Residência, foquei em definir exatamente em qual área eu iria me aprofundar. Então, decidi explorar os meus tópicos de interesse: Engenharia de Dados e Computação Distribuída. Portanto, decidi trabalhar com Processamento de Dados Massivos. Coletei alguns artigos que são muito importantes na área:

- MapReduce: Simplified Data Processing on Large Clusters
- The Hadoop Distributed File System
- Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing
- Spark: Cluster Computing with Working Sets

Além disso, fiz a leitura do primeiro capítulo do livro **"Fundamentos de Engenharia de Dados - Joe Reis".** O meu objetivo com essa leitura é ter uma noção holística sobre a área.
Resumo - Capítulo 1

Por fim, pesquisei alguns surveys na área:

- The Family of MapReduce and Large Scale Data Processing Systems
- Big Data Engineering and Distributed Systems Integration: Perspective (2023)

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

- Leitura e entendimento dos artigos e surveys coletados
- Ler o capítulo livro Fundamentos de Engenharia de Dados que fornece informações a respeito da parte de Processamento de Dados
- Procurar mais informações sobre sistemas distribuídos

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

Resumo - Capítulo 1. Documento citado no Termo de Aceite de Entrega do dia 4 de setembro de 2025

Resumo - Fundamentos de Engenharia de Dados Capítulo 1

O primeiro capítulo do livro apresenta diversas definições de Engenharia de Dados. A mais relevante é a que descreve a área como responsável por planejar, construir e manter todo o ciclo de vida dos dados, abrangendo coleta, ingestão, armazenamento, processamento e disponibilização para consumo por analistas, cientistas de dados e engenheiros de machine learning.

O autor também destaca áreas que atravessam todas essas etapas, como **Segurança e Privacidade**, **Qualidade e Governança de Dados**, **DataOps e Orquestração** e **Engenharia de Software aplicada a dados**.

Outro ponto importante é a introdução ao conceito de arquitetura de dados, apresentada como essencial para dar coerência ao fluxo de informações e facilitar a integração entre sistemas. Uma boa arquitetura, segundo o livro, serve de base para a escolha das ferramentas e tecnologias utilizadas ao longo de todo o ciclo.

Por fim, o capítulo contextualiza a evolução da profissão de engenheiro de dados, mostrando como ela se consolidou com o boom do Big Data. Antes, as tarefas ligadas a dados ficavam dispersas entre DBAs, engenheiros de software e analistas de BI, em processos de ETL voltados a data warehouses relacionais. O aumento do volume e da complexidade dos dados — impulsionado por redes sociais, IoT e aplicações web — demandou novas soluções de escalabilidade e processamento distribuído, levando ao surgimento de ferramentas como Hadoop, Spark e Kafka e à necessidade de profissionais especializados em projetar e manter pipelines modernos.

Esse capítulo funciona como uma introdução geral aos conceitos, preparando o terreno para discussões mais profundas nos capítulos seguintes.

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: **11 de set. de 2025**

Participantes da Entrega **[matriculados em Residência em IA]:**

MARCELO HENRIQUE ALVES PEREIRA SOBRINHO

Entrega: **[descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]**

Para esta Semana da Residência, continuei aprofundando os meus estudos na área de Computação Distribuída e Processamento Massivo de Dados.

Realizei a leitura e resumo dos dois artigos:

- Spark: Cluster Computing with Working Sets
  - Apache Spark - Considerações sobre o artigo
- MapReduce: Simplified Data Processing on Large Clusters:
  - Resumo MapReduce

Os conhecimentos adquiridos a partir desses dois artigos foram fundamentais para compreender arquiteturas que marcaram a evolução da área em que pretendo atuar, além de servirem como base para modelos mais modernos. Ademais, um dos artigos já introduz conceitos de Machine Learning.

Além disso, meu objetivo foi direcionar os estudos para aplicações de Inteligência Artificial. Para isso, com o apoio do Gemini, busquei fontes de informação sobre Computação Distribuída aplicada especificamente à IA. Nesse processo, me foi fornecido o survey ***"A Survey on Distributed Machine Learning"***, que apresenta de forma abrangente os principais tópicos e fundamentos dessas duas áreas de uma maneira geral

3 A Survey on Distributed Machine Learning

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

- Ler o survey citado anteriormente
- Buscar mais informações sobre treinamento e inferência distribuída de algoritmos de Machine Learning

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

O meu tema ainda não está 100% definido, mas creio que com a leitura desse survey eu já terei o direcionamento adequado.

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Apache Spark - Considerações sobre o artigo. Documento citado no Termo de Aceite de Entrega do dia 11 de setembro de 2025

Considerações sobre o artigo oficial do Apache Spark

Abstract

MapReduce and its variants have been highly successful in implementing large-scale data-intensive applications on commodity clusters. However, most of these systems are built around an acyclic data flow model that is not suitable for other popular applications. This paper focuses on one such class of applications: those that reuse a working set of data across multiple parallel operations. This includes many iterative machine learning algorithms, as well as interactive data analysis tools. We propose a new framework called Spark that supports these applications while retaining the scalability and fault tolerance of MapReduce. To achieve these goals, Spark introduces an abstraction called resilient distributed datasets (RDDs). An RDD is a read-only collection of objects partitioned across a set of machines that can be rebuilt if a partition is lost. Spark can outperform Hadoop by 10x in iterative machine learning jobs, and can be used to interactively query a 39 GB dataset with sub-second response time.

O artigo inicialmente argumenta que sistemas como o Hadoop (MapReduce) são bons para processar dados em uma única passagem, mas muito ineficientes para tarefas que precisam reutilizar os mesmos dados várias vezes, como em algoritmos machine learning e análises de dados interativas. O Spark foi criado para resolver exatamente esse problema, mantendo os dados em memória para acesso rápido e repetido utilizando um novo conceito chamado de Resilient Distributed Datasets (RDDs).

O problema do MapReduce é que para cada nova tarefa ou cálculo, todos os dados devem ser lidos novamente em disco, o que é uma operação lenta se você quisesse, por exemplo, treinar um modelo de Machine Learning. Os RDDs são uma abstração que conseguem solucionar esse problema: uma coleção de dados que é particionada e dividida entre várias máquinas em um cluster. O grande diferencial, é que um RDD pode ser mantido na memória RAM do computador. Com isso, os dados podem ser acessados de maneira muito mais rápida que no disco rígido.

Resilient Distributed Dataset

A principal abstração do Spark é o RDD. Eles representam uma coleção read-only de objetos particionados em várias máquinas de um cluster. Os RDDs conseguem ser reconstruídos se houver falha em uma das máquinas ou na própria partição. A reconstrução acontece processando novamente todas as transformações que ocorreram no RDD desde a sua origem até a falha.

Coleção de Objetos Particionada e Distribuída: Um RDD não é apenas uma lista de dados, mas uma lista dividida em partições. Cada partição pode ser enviada para uma máquina diferente no cluster. Isso é o que permite ao Spark realizar operações em paralelo: cada nó trabalha em sua própria partição de dados simultaneamente.

Imutável (read-only)**:** O RDD nunca é “modificado”. Em vez disso, é aplicada uma transformação a ele, que como resultado cria um novo RDD. Por exemplo, se você filtrar um RDD, o resultado não é o RDD original com menos itens, mas sim um RDD completamente novo que contém apenas os itens filtrados. Essa imutabilidade elimina a complexidade de garantir que as atualizações de dados sejam consistentes em todo o cluster.

Resiliente**:** Um RDD sempre se “recorda” de como foi criado. Ele mantém um registro de todas as transformações que o geraram a partir de sua fonte original. Esse histórico de transformações é chamado de Grafo de Linhagem ou DAG.

Um RDD pode ser criado de quatro maneiras:

- Lendo um arquivo de um sistema de armazenamento distribuído, como o HDFS

- Pegando uma estrutura de dados (uma lista por exemplo) que existe no programa principal e distribuindo-a pelo cluster para transformá-la em um RDD.
- Transformando um RDD existente. Como foi dito anteriormente, podemos aplicar transformações em um RDD para gerar um novo.
- Alterando a persistência de um RDD.

Outra coisa que é importante destacar, é que os RDDs são “passageiros”. Após serem calculados para uma ação, eles são descartados da memória. Se você precisar usar o mesmo RDD novamente em outra ação, o Spark terá que refazer todas as transformações desde o início, com base no Grafo de Linhagem. Para algoritmos de machine learning, como foi falado antes, esse comportamento é um gargalo muito grande.
Por isso, o Spark permite que você altere esse comportamento, instruindo-o a manter um RDD na memória após a primeira vez que ele for calculado. Isso é feito através de comandos no programa principal. Ao colocar um RDD, na próxima vez que ele for necessário, o Spark o acessará diretamente da memória RAM, que é muito mais rápido do que recalcular tudo do zero.

É importante ressaltar que os comandos para realizar essas ações são uma otimização. Se o cluster não tiver memória suficiente para armazenar o RDD, o Spark não vai falhar; ele simplesmente vai recalcular as partições que não couberam na memória quando elas forem necessárias.

Exemplos de Uso

O artigo nos fornece três exemplos do uso do Spark: Busca textual, Regressão Logística e Algoritmo dos Mínimos Quadrados Alternados. Para o contexto da residência, os dois últimos são os mais importantes, mas decidi olhar com mais detalhes o de **Regressão Logística**.

```
// Read points from a text file and cache them
val points = spark.textFile(...)
.map(parsePoint).cache()
// Initialize w to random D-dimensional vector
var w = Vector.random(D)
// Run multiple iterations to update w
for (i <- 1 to ITERATIONS) {
val grad = spark.accumulator(new Vector(D))
for (p <- points) { // Runs in parallel
val s = (1/(1+exp(-p.y*(w dot p.x)))-1)*p.y
grad += s * p.x
}
w -= grad.valu
```

Entrarei em mais detalhes em uma versão própria desse algoritmo na próxima semana da Residência.

Implementação do Spark

Sem entrar em detalhes aqui, mas é importante comentar sobre. O Spark não foi construído do zero para gerenciar a parte dos clusters. Em vez disso, ele foi construído sobre o Mesos, que é como um SO para clusters.

O Mesos basicamente gerencia os recursos de todas as máquinas do cluster (CPU, Memória) e permite que diferente aplicações (Spark, Hadoop) compartilhem esses recursos de forma eficiente. Assim, não precisamos gerenciar todos esses recursos de baixo nível.

Resultados

Os experimentos realizados pelos pesquisadores provam que o Spark é muito mais rápido que o Hadoop MapReduce para tarefas iterativas e interativas, devido à sua capacidade de manter os dados em memória.

O primeiro teste foi no algoritmo iterativo de regressão logística, onde utilizaram um conjunto de dados de aproximadamente 29 GB.

- Hadoop MapReduce: Cada iteração levou cerca de 127 segundos.

- Spark: A primeira iteração foi lenta, levando 174 segundos. No entanto, as iterações seguintes levaram apenas 6 segundos, já que os dados já estavam na memória

Os pesquisadores também simularam a falha de um nó durante a execução. O Spark conseguiu se recuperar e terminar a tarefa.

Resumo MapReduce. Documento citado no Termo de Aceite de Entrega do dia 11 de setembro de 2025

Resumo - MapReduce

A solução proposta pelos dois autores deste artigo foi uma nova interface que permite que os programadores expressem as operações simples que desejam realizar, enquanto uma biblioteca de tempo de execução esconde os detalhes complexos da computação distribuída. A interface foi inspirada nas primitivas funções ***`map`*** e ***`reduce`*** de linguagens de programação funcionais como Scala. O modelo permite que os programas sejam automaticamente paralelizados e executados em grandes clusters de máquinas comuns, tornando a computação em larga escala acessível a programadores sem experiência em sistemas distribuídos.

O núcleo do *MapReduce* é o processamento de pares de chave/valor. O usuário define duas funções principais:

1. Função *`Map`*: Essa é uma função que é escrita pelo próprio programador. Ela processa um par de chave/valor de entrada e gera um conjunto de pares de chave/valor intermediários. Por exemplo, para contar palavras em um documento, a chave de entrada pode ser o nome do documento e o valor, seu conteúdo. A função *`map`* então emitiria um par **`(palavra, "1")`** para cada palavra encontrada no texto.
2. Função *`Reduce`*: Também escrita pelo usuário/programador, essa função é chamada após a fase de *`map`*. O sistema *MapReduce* agrupa todos os valores intermediários que possuem a mesma chave e os entrega à função *`reduce`*. A função *`reduce`* então mescla (ou "reduz") esses valores para formar um conjunto menor de valores, geralmente produzindo uma única saída. No exemplo da contagem de palavras, para a chave "the", a função

`reduce` receberia uma lista de "1"s (ex: `["1", "1", "1", ...]`) e somaria todos para emitir o resultado final `("the", 587)`.

O *MapReduce* é flexível e pode ser aplicado a uma vasta gama de problemas, como:

- **Distributed Grep:** Procurar por um padrão em um grande volume de texto.
- **Contagem de Frequência de Acesso a URLs:** Processar logs de servidores web para contar quantas vezes cada URL foi acessada.
- **Gráfico de Links da Web Invertido:** Criar um índice que, para cada URL de destino, lista todas as páginas de origem que apontam para ela.
- **Índice Invertido:** A base dos motores de busca, onde para cada palavra, é gerada uma lista de todos os documentos em que ela aparece.

A implementação do Google foi projetada para rodar em grandes clusters de *PCs* comuns (*commodity hardware*) conectados por Ethernet, um ambiente onde falhas de máquinas são frequentes. Quando o artigo cita *PCs* comuns, ele quer dizer computadores com hardwares acessíveis à população.

Fluxo de Execução

O processo de execução de um trabalho *MapReduce* segue os seguintes passos:

1. **Divisão da Entrada:** A biblioteca *MapReduce* primeiro divide os arquivos de entrada em `M` pedaços (*splits*), tipicamente de 16 a 64 MB cada.
    a. Detalhes:
        i. Cada *split* é processado individualmente por cada *worker* de map;
        ii. Normalmente os *splits* correspondem a blocos do HDFS (no *Hadoop*);
        iii. Quanto maior o tamanho dos *splits*, menor será o paralelismo; quanto menores os splits, maior será o paralelismo.
2. **Coordenação *Master-Worker*:** Uma cópia do programa do usuário atua como o **master**, enquanto as outras são ***workers***. O *master* é responsável por atribuir as
   `M` tarefas de *`map`* e `R` tarefas de *`reduce`* aos *workers* ociosos.

3. **Fase de Map:** Um *worker* que recebe uma tarefa `map` lê seu *split* de entrada correspondente, executa a função `map` definida pelo usuário e armazena os pares de chave/valor intermediários em um *buffer* na memória.
4. **Particionamento e Armazenamento Local:** Periodicamente, os dados intermediários em buffer são escritos no disco local do *worker*, particionados em R regiões usando uma função de particionamento. As localizações desses arquivos são enviadas de volta ao nó *master*.
    a. **Curiosidade**:
        i. A necessidade de escrever esses dados intermediários no disco é um dos principais gargalos que o *Spark* foi projetado para resolver. Em trabalhos iterativos, o *MapReduce* precisa recarregar os dados do disco a cada passo, resultando em uma penalidade de desempenho significativa. O *Spark*, por outro lado, introduziu a abstração de *RDDs* que podem ser mantidos em memória entre as operações, evitando essas custosas operações de leitura de disco.
5. **Fase de Shuffle (Embaralhamento):** Um worker que recebe uma tarefa `reduce` é notificado pelo master sobre as localizações dos dados intermediários. Ele então lê esses dados dos discos locais dos *workers* de `map` via chamadas de procedimento remoto. Após ler todos os dados, ele os ordena pela chave intermediária para agrupar todas as ocorrências da mesma chave.
6. **Fase de Reduce:** O worker de `reduce` itera sobre os dados ordenados e, para cada chave única, passa a chave e a lista de valores correspondentes para a função `reduce` do usuário. A saída da função `reduce` é anexada a um arquivo de saída final para aquela partição.
7. **Conclusão:** Quando todas as tarefas de `map` e `reduce` terminam, o master acorda o programa principal do usuário, e a chamada *MapReduce* retorna.

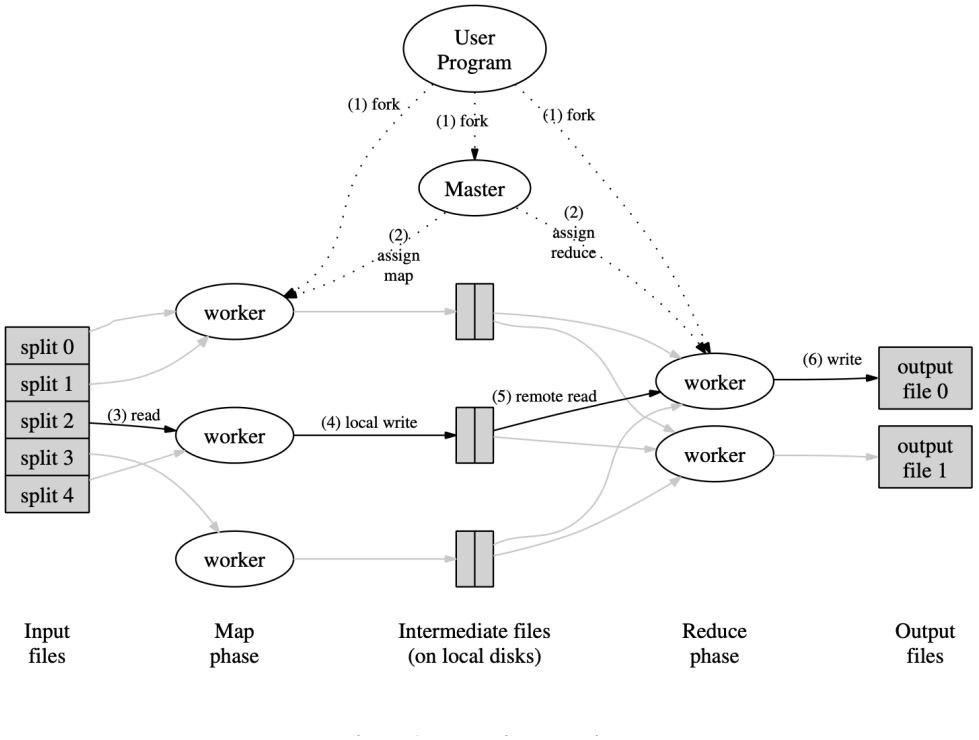

Figura 1: Exemplo do fluxo de execução do MapReduce em um ambiente de computação distribuída

Tolerância a Falhas

- **Falha de *Worker*:** O master monitora os workers com pings periódicos. Se um worker falha, todas as tarefas que ele estava executando (ou já havia completado) são redefinidas para o estado "ocioso" e reatribuídas a outros workers. As tarefas `map` concluídas precisam ser reexecutadas porque sua saída estava no disco local da máquina que falhou e se tornou inacessível. As tarefas `reduce` concluídas não precisam ser reexecutadas, pois sua saída é armazenada em um sistema de arquivos global e replicado (como o *Google File System - GFS*).
- **Falha do *Master*:** A implementação atual aborta a computação se o master falhar, pois ele é um ponto único de falha. No entanto, como a falha de um

único master é improvável, essa abordagem foi considerada aceitável, e os clientes podem simplesmente tentar rodar o trabalho novamente.

Otimização de Localidade Para economizar a largura de banda da rede, que é um recurso escasso, o master do MapReduce tenta agendar as tarefas

map em máquinas que já contêm uma réplica dos dados de entrada em seus discos locais. Em grandes execuções, a maioria dos dados de entrada é lida localmente, sem consumir banda de rede.

Tarefas de Backup (Backup Tasks) Para mitigar o problema de "stragglers" (máquinas que demoram muito para concluir uma das últimas tarefas), o sistema emprega um mecanismo de tarefas de backup. Perto do final do trabalho, o master agenda execuções de backup para as tarefas ainda em andamento. A tarefa é marcada como concluída assim que a execução primária ou a de backup terminar. Isso reduz significativamente o tempo total de conclusão, com um pequeno aumento no uso de recursos computacionais.

Refinamentos e Recursos Adicionais

- **Função de Particionamento:** Os usuários podem fornecer uma função de particionamento personalizada para controlar como as chaves intermediárias são distribuídas entre os *reducers*. Por exemplo, para garantir que todas as URLs do mesmo host sejam processadas pelo mesmo *reducer*.
- **Função Combiner:** Uma função *combiner* opcional pode ser especificada para realizar uma pré-agregação de dados no final da fase de *map*, antes que os dados sejam enviados pela rede. Isso é muito eficaz quando a função *reduce* é comutativa e associativa (como uma soma), reduzindo drasticamente o tráfego de rede.
- **Contadores:** Uma API simples para que os programas do usuário possam contar eventos de interesse (ex: "documentos corrompidos" ou "palavras em alemão indexadas"). Os valores são agregados pelo *master* e retornados ao final do trabalho.

Desempenho

Os testes foram realizados em um cluster de aproximadamente 1800 máquinas.

- Grep: Uma busca por um padrão raro em 1 TB de dados (10^{10}) registros de 100 *bytes*) levou cerca de 150 segundos do início ao fim, atingindo um pico de taxa de leitura de entrada de mais de 30 GB/s.
- Sort: A ordenação de 1 TB de dados (10^{10} registros de 100 bytes) levou 891 segundos (cerca de 15 minutos). O desempenho foi fortemente beneficiado pelas otimizações:
    - Sem tarefas de backup**:** A mesma tarefa de ordenação levou 1283 segundos (um aumento de 44%) devido a alguns "*stragglers*" no final.
    - **Com falha de máquinas:** Ao matar intencionalmente 200 processos de *workers* no meio da execução, o trabalho foi concluído em 933 segundos (apenas 5% mais lento), demonstrando a resiliência do sistema

Conclusão e Considerações Finais

Por fim, gostaria de deixar alguns comentários acerca deste artigo. Embora o Hadoop MapReduce tenha sido substituído pelo Apache Spark para tarefas massivas, ele ainda desempenha um papel muito importante para entendermos alguns fundamentos de computação distribuída; além, claro, de que o Spark reaproveitou bastante coisa do MapReduce. Portanto, a leitura deste artigo ainda é bastante válida para quem quer entender um pouco mais sobre os fundamentos dessa área.

APÊNDICE 2

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: **18 de set. de 2025**

Participantes da Entrega **[matriculados em Residência em IA]**:

MARCELO HENRIQUE ALVES PEREIRA SOBRINHO

Entrega: **[descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]**

Na terceira Semana da Residência em IA, continuei aprofundando meus estudos na intersecção das áreas de Computação Distribuída, Processamento de Dados e Machine Learning. Como parte do planejamento da semana anterior, o meu intuito foi fazer a leitura do Survey: **"A Survey on Distributed Machine Learning".**

Fiz anotações a respeito dos pontos que achei mais interessantes desse Survey, que se encontram no seguinte documento: Anotações - A Survey on Distributed Machine Learning

Um dos temas abordados na seção de privacidade de dados do Survey é o **Aprendizado Federado**. Esse assunto despertou minha curiosidade e, por isso, decidi me aprofundar um pouco mais. Fiz uma leitura inicial sobre a técnica e percebi o quanto ela é interessante, o que me motivou a escolhê-la como tema principal do meu estudo. Para complementar minha pesquisa, utilizei o **Gemini** com a ferramenta de **Deep Research** para localizar artigos relacionados ao tema.

A descentralização do treinamento de modelos de machine learning, abordada no *Survey*, me despertou muito interesse e foi a principal motivação para a escolha do Aprendizado Federado como minha área de estudo.

Encontrei o artigo seminal sobre **Aprendizado Federado**, intitulado ***"Communication-Efficient Learning of Deep Networks from Decentralized Data"*** **(2017), de H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson e Blaise Agüera y Arcas**. Além dele, selecionei também o trabalho ***"Federated Learning: Challenges, Methods, and Future Directions"*** **(2020), de Tian Li, Anit Kumar Sahu, Ameet Talwalkar e Virginia Smith**, por considerar relevante para a análise dos principais desafios e perspectivas dessa área.

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

- Realizar a leitura dos dois artigos para ter uma noção melhor sobre o assunto e me aprofundar mais.
- Desenvolver um diagrama das arquiteturas de Aprendizado Federado aplicadas em cenários reais

- Explorar *frameworks* relacionados ao assunto
- Procurar artigos mais recentes sobre o tema (entre 2023-2025)

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Anotações - A Survey on Distributed Machine Learning. Documento citado no Termo de Aceite de Entrega do dia 18 de setembro de 2025

Introdução

O Survey começa destacando que a demanda por IA e Machine Learning cresceu bastante nos últimos tempos. Porém, para treinar modelos maiores e mais complexos, como redes neurais profundas, a quantidade de dados necessária cresceu exponencialmente. Como o poder individual das máquinas não acompanhou esse crescimento, foi necessário utilizar artifícios da computação distribuída para realizar esse trabalho, como distribuir a carga de trabalho entre várias máquinas em um cluster. Isso introduz desafios, principalmente na paralelização eficiente do processo de treinamento. O Survey oferece uma visão geral sobre as aplicações de Computação Distribuída em Machine Learning. Eu direcionei o meu foco para duas seções deste Survey, que são descritas abaixo.

Topologias de sistemas

Esta seção e próxima, ao meu ver, abordam os temas mais pertinentes para mim. O primeiro parágrafo aborda o conceito de topologia para esse contexto, que consiste em definir como os diferentes computadores de um cluster são organizados e como eles trocam informações para treinar um modelo de Machine Learning de forma colaborativa. A escolha da topologia impacta diretamente a performance, a escalabilidade e a tolerância a falhas do sistema.

São definidas três categorias de topologia:

- **Centralizada (Ensembling)**
  - Há um nó central que distribui as tarefas para os nós de trabalho. Cada nó de trabalho treina seu próprio modelo de forma independente, usando uma parte dos dados. Ao final, eles enviam seus resultados (pesos, no caso de ANNs) para o nó central, que tem a responsabilidade de agregar tudo em um único modelo final coeso. É uma arquitetura simples de implementar e gerenciar, porém, se o nó central falhar, todo o processo falha; isso pode ser um gargalo, pois toda a informação precisa passar por ele em algum momento.
- **Descentralizada**

- Neste modelo, não há um único nó central. A responsabilidade é compartilhada, permitindo agregações intermediárias, o que melhora a escalabilidade e a tolerância a falhas. Existem duas variações populares:
  - **Árvore** (usada no paradigma AllReduce):
    - Os nós são organizados em uma estrutura de árvore. O processo todo ocorre em duas fases:
      - **Fase de Agregação (Reduce):** Cada nó folha calcula sua atualização local (gradiente por exemplo). Em seguida, ele passa essa atualização para o nó pai na árvore. O nó pai combina sua própria atualização com as que recebeu de seus nós filhos e passa o resultado agregado para o nível acima. Isso continua até que o nó raiz receba a agregação global de todos os nós.
      - **Fase de Distribuição:** Uma vez que o nó raiz tem o resultado global, ele o envia de volta para seus filhos, que por sua vez vão repassando para os seus, e assim por diante, até que todos os nós na árvore tenham o mesmo resultado final atualizado.
    - É uma forma muito boa de calcular uma soma global e distribuir o resultado, minimizando o tráfego de rede em comparação com um modelo centralizado.
  - **Parameter Server**
    - A arquitetura é dividida entre os nós de trabalho e os servidores de parâmetros. O estado global do modelo é dividido e armazenado nos servidores de parâmetros. Os nós de trabalho são responsáveis por realizar o treinamento: eles solicitam os parâmetros mais atualizados dos servidores, calculam as atualizações com base em seus dados e enviam essas atualizações de volta para os servidores, que aplicam a mudança no modelo global. Assim como na arquitetura centralizada, os servidores de parâmetros podem ser um gargalo de

comunicaçlão, já que precisam lidar com todas as solicitações de leitura e escrita dos nós de trabalho.

- **Totalmente Distribuída (Peer-to-Peer)**
  - Eu achei essa a forma mais interessante de descentralização. Não há nós trabalhadores como em outras arquiteturas; todos eles são iguais (peers). Funciona igual quando baixamos um arquivo no Torrent. Cada nó mantém sua própria cópia do modelo e se comunica diretamente com outros nós para sincronizar as atualizações. É um modelo bastante escalável, mas é difícil garantir a convergência de todos os nós para o mesmo modelo final.

Modelos de Comunicação

O modelo de comunicação dita quando e como os nós sincronizam as suas informações. Há um trade-off nessa questão: sincronizar os dados frequentemente garante que todos trabalhem com a versão mais atualizada do modelo, mas pode causar atrasos. Sincronizar com menos frequência acelera o processo, mas os nós podem trabalhar com versões desatualizadas do modelo e isso pode prejudicar a convergência. Existem alguns que são comentados:

- **Bulk Synchronous Parallel**: Nesse modelo, a cada passo, todos os nós primeiro realizam seus cálculos de forma paralela. Depois que todos terminam, eles se comunicam para trocar as suas atualizações. Nenhum nó pode começar o próximo passo sem que a fase de comunicação esteja completa e todos os nós estejam sincronizados.
- **Stale Synchronous Parallel**: Esse modelo permite que os nós mais rápidos avancem um número limitado de passos à frente dos nós mais lentos. Em vez de esperar em cada passo, um nó pode continuar trabalhando com uma versão desatualizada dos parâmetros modelo que ele tem em cache. Há um limite máximo de atraso permitido. Se um nó estiver muito atrás, o sistema força uma sincronização imediata.

- **Asynchronous Parallel**: É uma espécie de “cada um por si”. Não há barreiras de sincronização. Cada nó opera de maneira independente do outro. Os nós pegam a versão mais recente dos parâmetros do modelo sempre que precisam, calculam sua atualização e as envia de volta imediatamente, sem esperar por nenhum outro nó.

Por fim, uma coisa que também achei muito interessante foi a seção que fala a respeito da privacidade de dados nesse contexto. É falado sobre o Aprendizado Federado, que consiste em uma arquitetura que permite que múltiplas partes treinem um modelo de rede neural de forma conjunta, mantendo os dados em seu local de origem.

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: **24 de set. de 2025**

Participantes da Entrega **[matriculados em Residência em IA]:**

MARCELO HENRIQUE ALVES PEREIRA SOBRINHO

Entrega: **[descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]**

Nessa Semana da Residência em IA, continuei explorando o conceito de **Aprendizado Federado**, no qual realizei a leitura de dois artigos na área:

- Communication-Efficient Learning of Deep Networks from Decentralized Data
- Federated Learning: Challenges, Methods, and Future Directions

Redigi o resumo destes dois artigos Resumo - Artigos sobre aprendizado federado

Também realizei a busca a respeito de *Frameworks* disponíveis para realizar futuros testes e encontrei o **NVIDIA FLARE** e o **TensorFlowFederated (TFF).** O objetivo é testar o Aprendizado Federado utilizando esses frameworks para simular ambientes.

Além disso, eu queria pesquisar sobre algum artigo que comentasse sobre a questão do Aprendizado Federado em dados "non-IID". Fiz uma busca e achei o artigo de pesquisadores chineses: **"Federated Learning on Non-IID Data Silos: An Experimental Study"**

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

- Realizar a leitura do artigo **Federated Learning on Non-IID Data Silos: An Experimental Study**
- Ler a documentação dos dois frameworks e escolher um para seguir com os testes.

Como resultado, eu planejo:

- Escrever um resumo sobre o artigo.
- Criar um repositório no GitHub para começar a fazer os testes em código.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Resumo - Artigos sobre Aprendizado Federado. Documento citado no Termo de Aceite de Entrega do dia 24 de setembro de 2025

Resumo dos artigos: **Communication-Efficient Learning of Deep Networks from Decentralized Data[1] e Federated Learning: Challenges, Methods, and Future Directions[2]**

A Aprendizagem Federada é uma abordagem de aprendizado de máquina descentralizada projetada para treinar modelos a partir de dados distribuídos em dispositivos remotos, como celulares, ou em silos de dados, como hospitais. O princípio fundamental é que os dados de treinamento permanecem em sua localização original, preservando a privacidade e reduzindo os custos de comunicação. Um servidor central coordena o processo, aprendendo um modelo global compartilhado através da agregação de atualizações calculadas localmente pelos dispositivos (clientes).

O artigo [1] é fundamental, pois foi nele que a Google, em 2017, apresentou pela primeira vez o conceito de Aprendizagem Federada. Mais do que apenas introduzir a ideia, o trabalho descreve a topologia desse novo paradigma de treinamento e detalha uma série de testes e experimentos. Os autores demonstram de forma prática como essa abordagem descentralizada não só é viável, mas também eficiente, permitindo treinar modelos sem a necessidade de mover dados sensíveis.

Principais Desafios

Ambos os artigos destacam que a Aprendizagem Federada possui um conjunto único de desafios que a diferencia da otimização distribuída tradicional em data centers. Os principais desafios são:

- **Comunicação é custosa:** A comunicação é o principal gargalo, sendo muito mais lenta e cara do que a computação local nos dispositivos modernos. Portanto, o objetivo central é desenvolver métodos que sejam eficientes em comunicação, reduzindo o número total de rodadas de comunicação necessárias para treinar um modelo.

- **Heterogeneidade Estatística:** Os dados nos dispositivos são inerentemente não-IID (não independentes e identicamente distribuídos) e desbalanceados. Isso significa que os dados de cada cliente não são uma amostra representativa da população geral e que os clientes podem ter quantidades de dados muito diferentes. Essa característica viola as premissas comuns de muitos algoritmos de otimização distribuída.
- **Heterogeneidade de Sistemas:** Os dispositivos na rede variam drasticamente em termos de hardware (CPU, memória), conexão de rede (Wi-Fi, 4G) e disponibilidade de energia. Além disso, apenas uma pequena fração dos dispositivos pode estar ativa a qualquer momento, e eles podem abandonar o processo de treinamento inesperadamente.
- **Privacidade:** Embora manter os dados localmente seja um grande avanço em privacidade, a comunicação das atualizações do modelo (como gradientes) ainda pode revelar informações sensíveis dos usuários. São necessárias técnicas adicionais, como computação segura multipartidária (SMC) ou privacidade diferencial, para fornecer garantias de privacidade mais fortes.

Soluções e Métodos

O primeiro artigo introduz o **Federated Averaging (FedAvg)** como uma solução prática e eficaz para esses desafios. Em vez de aplicar uma abordagem ingênua de Gradiente Descendente Estocástico (SGD), onde apenas um pequeno cálculo é feito por rodada, o **FedAvg** reduz a comunicação ao aumentar a computação local. O processo funciona da seguinte forma:

1. Um servidor central envia o modelo atual para um subconjunto de clientes.
2. Cada cliente treina o modelo intensivamente com seus dados locais por várias etapas (épocas).
3. Os clientes enviam seus modelos atualizados (pesos) de volta ao servidor.
4. O servidor faz uma média ponderada dos modelos recebidos para produzir o novo modelo global.

Experimentos demonstram que o **FedAvg** é robusto aos dados não-IID e desbalanceados e pode reduzir o número de rodadas de comunicação necessárias em 10 a 100 vezes em comparação com o SGD sincronizado.

O segundo artigo, funcionando como uma pesquisa da área, contextualiza o **FedAvg** como o método de fato para a Aprendizagem Federada e explora outras abordagens para lidar com os desafios, como esquemas de compressão para reduzir o tamanho das mensagens , treinamento descentralizado sem um servidor central , e métodos de modelagem que criam modelos personalizados para cada dispositivo para lidar com a heterogeneidade estatística, como a aprendizagem multitarefa e a meta-aprendizagem.

Direções Futuras

Ambos os artigos sugerem que a Aprendizagem Federada é uma área de pesquisa ativa. O segundo artigo delineia direções futuras, incluindo o desenvolvimento de esquemas de comunicação mais radicais (como "one-shot learning") , a criação de diagnósticos para quantificar a heterogeneidade da rede, a exploração de problemas além da aprendizagem supervisionada, e a necessidade de benchmarks padronizados para garantir a reprodutibilidade e o avanço da área.

A Formulação Matemática da Aprendizagem Federada

É claro que não podemos deixar de fora a parte matemática comentada nos artigos. Nas subseções abaixo, eu descrevo com um pouco de detalhes como os autores pensaram e nos trouxeram o conhecimento de como essa arquitetura funciona na prática.

1. O Objetivo Global

O objetivo final em muitos problemas de aprendizado de máquina é encontrar os parâmetros de um modelo, **w**, que minimizem uma função de perda sobre um conjunto de dados:

$$\min_{w \in \mathbb{R}^d} f(w) \quad \text{onde} \quad f(w) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} f_i(w)$$

- **w**: vetor de parâmetros do modelo que está sendo treinado
- **n:** número total de amostras de dados em todos os clientes

- **fi(w):** função de perda calculada para uma única amostra de dado i. Ela mede o erro do modelo para aquele ponto de dado específico.

2. A Decomposição Federada

No cenário federado, os dados são particionados entre K clientes. A função de objetivo global é, portanto, reescrita como uma média ponderada das funções de perda locais de cada cliente.

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w)$$

- K: número total de clientes
- nk: número de amostras de dados no cliente k
- Fk(w): função de perda média local para o cliente k, calculada exclusivamente sobre seus próprios dados. Ela é definida como:

$$F_k(w) = \frac{1}{n_k} \sum_{i \in \mathcal{P}_k} f_i(w)$$

onde Pk é o conjunto de índices dos dados pertencentes ao cliente k.

3. A Atualização do Modelo no FedSGD

O algoritmo de base, **FederatedSGD (FedSGD)**, pode ser visto de duas maneiras matematicamente equivalentes.

- **Visão 1: Agregação de Gradientes** O servidor atualiza o modelo global dando um passo na direção oposta da média ponderada dos gradientes calculados pelos clientes.

$$w_{t+1} \leftarrow w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k$$

gk = ∇Fk(wt) é o gradiente médio do cliente k no modelo atual wt.

- **Visão 2: Agregação de Modelos** Alternativamente, cada cliente pode dar um passo de gradiente descendente localmente, e o servidor faz a média dos modelos resultantes.
  - **No cliente k**: wt+1k←wt−ηgk.
  - **No servidor**: wt+1←∑k=1Knnkwt+1k.

4. A Otimização no FedAvg

O **FedAvg** generaliza a segunda visão do **FedSGD**, permitindo que cada cliente execute mais de um passo de otimização localmente. Em vez de calcular o gradiente uma vez, cada cliente itera a atualização local múltiplas vezes.

- **Atualização Local Iterativa no Cliente k**:

$$w^k \leftarrow w^k - \eta \nabla F_k(w^k)$$

Este passo é repetido por E épocas locais em mini batches de tamanho B. Depois desse processo de treinamento local, o cliente envia o seu modelo final wkt+1 ao servidor, que então faz a média ponderada dos modelos, da mesma forma que na visão 2 do **FedSGD**.

APÊNDICE 3

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: **1 de out. de 2025**

Participantes da Entrega **[matriculados em Residência em IA]:**

MARCELO HENRIQUE ALVES PEREIRA SOBRINHO

Entrega: **[descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]**

Na Semana 5 da Residência em IA, continuei aprofundando os meus estudos sobre o tema: **Aprendizado Federado**

Realizei a leitura do artigo: Federated Learning on Non-IID Data Silos: An Experimental Study

Este artigo faz experimentos muito importantes para a área de Aprendizado Federado, onde os pesquisadores testaram quatro diferentes algoritmos federados (**FedProx, FedAvg, FedNova, SCAFFOLD**) em vários tipos de "desvio" de dados. Esses desvios são:

- Dados de usuários que diferem em quantidade (ex: cenário em que o tamanho do conjunto de dados local varia entre os usuários;
- Dados de usuários que diferem na distribuição de rótulos (ex: quando um usuário tem mais dados de uma classe específica do que outro usuário;
- Dados de usuários que diferem na distribuição das features (ex: quando vários usuários têm o mesmo mapa de features, porém as características dos dados de cada usuário são distintas).

Escrevi um resumo a respeito deste artigo, incluindo a explicação dos algoritmos:
Resumo - Federated Learning on Non-IID Data Silos: An Experimental Study

Além disso, pesquisei sobre os dois frameworks mencionados nas semanas anteriores: **TensorFlow Federated** e **NVIDIA FLARE**. Optei por seguir com o **NVIDIA FLARE**, principalmente por oferecer suporte ao **PyTorch**, framework com o qual já possuo experiência. Não gostaria de ficar restrito ao "ecossistema" do TensorFlow para o desenvolvimento das redes neurais. Após uma leitura inicial da documentação do NVIDIA FLARE, percebi que ele será um recurso valioso para aliar a teoria que venho estudando à prática.

Link da documentação: NVIDIA FLARE

Criei o repositório no GitHub onde vou colocar todos os códigos realizados com esse framework: https://github.com/marcelinhohaps/nvidia-flare-fl-res

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

- Pretendo aprofundar a leitura da documentação do NVIDIA-FLARE e iniciar a construção de uma arquitetura de ambiente federado para realizar testes. Em seguida, tenho interesse em disponibilizar essa arquitetura, juntamente com o código associado, em um repositório no GitHub.
- Revisar alguns conceitos de estatística (principalmente a parte formal de **distribuições de probabilidade**) que acabei esquecendo desde que cursei a disciplina, em 2021;
- Pesquisar sobre algoritmos mais recentes na área.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

O tema é muito legal

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Resumo - Federated Learning on Non-IID Data Silos: An Experimental Study.

Documento citado no Termo de Aceite de Entrega do dia 1 de outubro de 2025

Resumo sobre o artigo Federated Learning on Non-IID Data Silos: An Experimental Study

O artigo aborda um desafio central na era dos "silos de dados": como treinar modelos de machine learning eficazes quando os dados estão fragmentados em múltiplas fontes (organizações, países) e não podem ser centralizados devido a regulamentações de privacidade como a GDPR. O Aprendizado Federado (FL) surge como uma solução promissora, permitindo o treinamento colaborativo sem a troca de dados brutos. No entanto, a eficácia da FL é severamente prejudicada pelo fato de que os dados nesses silos são geralmente não independentes e identicamente distribuídos (não-IID), o que significa que cada usuário tem uma distribuição de dados diferente. Essa heterogeneidade atrapalha bastante a otimização dos modelos. Os autores argumentam que os algoritmos de FL existentes, embora promissores, careciam de uma avaliação sistemática e completa. Para resolver essa lacuna, o estudo propõe o NIID-Bench, um benchmark com estratégias abrangentes de particionamento de dados para simular cenários não-IID baseados na realidade.

Simulando Dados Não-IID

Aqui entra um dos pontos mais importantes dos artigos para mim. Para criar um ambiente de teste controlado, os pesquisadores focaram em simular três tipos principais de desvio (skew) de dados, particionando datasets públicos:

- **Desvio de distribuição de rótulos:** A distribuição de classes $P(y_i)$ varia entre as partes, e é modelado de duas formas:
  - Baseada em quantidade, onde cada usuário possui dados de apenas k classes
  - Baseada em distribuição $(p_k \sim \mathrm{Dir}(\beta))$, onde a proporção de cada classe por parte é determinada por uma distribuição de Dirichlet, com um parâmetro β controlando o nível de desequilíbrio
- **Desvio de distribuição de atributos:** A distribuição dos dados de entrada $P(x_i)$ difere entre as partes. Isso é simulado adicionando ruído Gaussiano com variâncias

distintas aos dados de cada parte ou usando dados do mundo real, como particionar o dataset FEMNIST, já que cada um possui um estilo de escrita única

- **Desvio de quantidade:** Quando o tamanho do dataset varia entre as partes, o que também é simulado com uma distribuição de Dirichlet para alocar diferentes volumes de dados a cada participante.

Algoritmos para lidar com a heterogeneidade dos dados

O problema central causado pelos dados não-iid é o “client drift”, onde os modelos locais, ao serem otimizados para seus dados específicos, se afastam do ótimo global que o modelo federado tenta alcançar. O artigo avalia três algoritmos projetados para reduzir esse problema, além do FedAvg. Recapitulando um pouco do FedAvg, temos a sua equação:

$$w^{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_k^{t+1}$$

Agregação do FedAvg.

w^{t+1}: pesos globais do modelo na rodada $t+1$;

K: número total de clientes participantes na rodada;

n_k: número de amostras do cliente k;

$n = \sum_{k=1}^{K} n_k$: número total de amostras em todos os clientes;

w_k^{t+1}: pesos do modelo do cliente k após o treinamento local na rodada $t+1$.

- **FedProx:** Adiciona um termo de regularização à função de perda local para limitar o quão distante o modelo local ω pode se afastar do modelo global recebido ω^t. A sua função de perda modificada é:

$$L(w;b) = \sum l(w;x;y) + \frac{\mu}{2}\|w - w^t\|^2$$, onde

Loss do FedProx.

- $L(w;b)$ é a função de perda total do cliente, parametrizada pelo modelo local w e por um batch b;
- $\sum l(w;x;y)$ é a soma ou a média das perdas individuais calculadas sobre o conjunto de dados locais do cliente;
- o termo $\frac{\mu}{2}\|w - w^t\|^2$ penaliza a distância Euclidiana entre os modelos, com a força da penalidade controlada pelo hiperparâmetro μ .

- **SCAFFOLD:** Utiliza uma técnica de redução de variância para corrigir o gradiente **local** em cada passo de otimização. Ele introduz variáveis de controle no servidor (c) e em cada cliente (c_i) para estimar a direção do gradiente global e local, respectivamente. A atualização local é então corrigida pela diferença entre essas direções:

$$w_i^{t+1} \leftarrow w_i^t - \eta \left(\nabla L(w_i^t; b) - c_i^t + c\right)$$

SCAFFOLD.

- c_i^t é a variável de controle do cliente. É uma estimativa da “direção de atualização” média do cliente i;
- c é a variável de controle do servidor. É uma estimativa da “direção de atualização” do modelo global;
- O cliente não usa seu gradiente puro. Ele o corrige;

- **FedNova:** Aborda a inconsistência na agregação quando os clientes realizam diferentes números de atualizações locais (τ_i). Esse algoritmo não altera o treinamento local, mas sim a forma como o servidor agrega as atualizações dos clientes no final da rodada. Em vez de fazer a média das atualizações totais, ele normaliza a atualização de cada cliente pelo seu número de passos ($g_i^t = \Delta w_i^t / \tau_i$),

agrega essas contribuições normalizadas e, em seguida, escala o resultado final por um número efetivo de passos, garantindo uma agregação mais justa.

- τ representa a quantidade de vezes que o gradiente foi calculado e o modelo foi atualizado localmente (número de batches processados para ser mais exato);
- $\Delta w_i^t = w^t - w_i^t$;
- Isso representa a mudança média por passo de atualização. Um cliente que treinou por muitas etapas terá uma grande atualização total. Ao dividir por τ, colocamos sua contribuição na mesma escala de um cliente que treinou por poucas etapas.

No pseudocódigo da próxima página, é mostrado o fluxo de implementação do FedNova, FedProx e FedAvg. As partes marcadas em laranja são referentes ao FedNova e as marcadas em vermelho são do FedProx.

Algorithm 1: A summary of FL algorithms including FedAvg/FedProx/FedNova. We use red and orange colors to mark the part specially included in FedProx and FedNova, respectively.

Input: local datasets $\mathcal{D}^i$, number of parties N, number of communication rounds T, number of local epochs E, learning rate η

Output: The final model w^T

Server executes:
initialize x^0
for $t = 0, 1, ..., T-1$ **do**
  Sample a set of parties S_t
  $n \leftarrow \sum_{i \in S_t} |\mathcal{D}^i|$
  for $i \in S_t$ **in parallel do**
    send the global model w^t to party P_i
    $\Delta w_i^t, \tau_i \leftarrow$ **LocalTraining**(i, w^t)
  For FedAvg/FedProx:
  $w^{t+1} \leftarrow w^t - \eta \sum_{i \in S_t} \frac{|\mathcal{D}^i|}{n} \Delta w_k^t$
  For FedNova:
  $w^{t+1} \leftarrow w^t - \eta \frac{\sum_{i \in S_t} |\mathcal{D}^i| \tau_i}{n} \sum_{i \in S_t} \frac{|D^i| \Delta w_i^t}{n \tau_i}$
return w^T

Party executes:
For FedAvg/FedNova: $L(w; \mathbf{b}) = \sum_{(x,y) \in \mathbf{b}} \ell(w; x; y)$
For FedProx:
$L(w; \mathbf{b}) = \sum_{(x,y) \in \mathbf{b}} \ell(w; x; y) + \frac{\mu}{2} \|w - w^t\|^2$
LocalTraining(i, w^t):
$w_i^t \leftarrow w^t$
$\tau_i \leftarrow 0$
for epoch $k = 1, 2, ..., E$ **do**
  for each batch $\mathbf{b} = \{\mathbf{x}, y\}$ of $\mathcal{D}^i$ **do**
    $w_i^t \leftarrow w_i^t - \eta \nabla L(w_i^t; \mathbf{b})$
    $\tau_i \leftarrow \tau_i + 1$
$\Delta w_i^t \leftarrow w^t - w_i^t$
return Δw_i^t, τ_i to the server

Pseudocódigo do treinamento federado.

Resultados

Antes de apresentar os resultados, os pesquisadores fornecem um resumo detalhado da configuração experimental:

- Conjunto de dados: Foram utilizados nove conjuntos de dados públicos, incluindo seis de imagem (MNIST, CIFAR-10, FMNIST, SVHN, FCUBE, FEMNIST) e três tabulares (adult, rcvl, covtype);
- Modelos de ML: CNN para os conjuntos de dados de imagem e MLP com três camadas ocultas para os dados tabulares;
- Parâmetros padrão: 10 participantes (exceto para o FCUBE, com 4), otimizador SGD com momentum 0.9, learning rate de 0.01 (0.1 para rcv1), batch size de 64 e 10 épocas locais por rodada de comunicação. Ocorreram 50 rodadas de comunicação no total;
- Métricas de avaliação: A principal métrica para comparação foi a acurácia top-1 no conjunto de teste.

Configurações não-IID

O desvio de distribuição de rótulos é o mais desafiador, causando a maior queda de acurácia em todos os algoritmos. O caso extremo, onde cada participante possui dados de uma única classe, apresenta o pior desempenho. As assimetrias na distribuição de características e de quantidade têm um impacto bem menor na acurácia. O FedAvg consegue lidar bem com a assimetria de quantidade devido ao seu cálculo da média ponderada baseada no número de amostras do cliente.

Algoritmos

Nenhum algoritmo é consistentemente superior em todos os cenários. O FedProx alcança a melhor acurácia em casos de assimetria de rótulo e quantidade. O SCAFFOLD tende a ter o melhor desempenho em assimetria de distribuição de características. A acurácia do SCAFFOLD é bastante instável; ele até consegue superar os outros em alguns

casos, mas também pode desempenhar mal em outros. O FedNova não demonstrou grande superioridade em comparação com os outros algoritmos.

Tarefas

Tarefas com os conjuntos de dados CIFAR-10 e os tabulares são particularmente desafiadoras em configuração não-IID. O MNIST é considerado uma tarefa simples, na qual todos os algoritmos estudados perfomam bem, mesmo em cenários não-IID.

Eficiência de comunicação

O FedProx tem uma velocidade de convergência semelhante ao FedAvg. Ambos necessitam de um número parecido de rodadas para chegar a um resultado, tendo, assim, uma eficiência de comunicação comparável. SCAFFOLD e FedNova são mais instáveis; a acurácia deles flutua muito entre as rodadas. Um treinamento instável pode exigir mais rodadas para garantir que o modelo atinja e mantenha uma boa performance, o que o torna menos eficiente em termos de comunicação.

Robustez às Atualizações Locais

O número de épocas locais tem um grande impacto na acurácia dos algoritmos. O valor ótimo para o número de épocas locais é altamente sensível à distribuição não-IId específica, indicando que os algoritmos existentes não são robustos a um grande número de atualizações locais. Por exemplo, com 80 épocas locais, a acurácia de todos os algoritmos se degradou significativamente no cenário de assimetria de distribuição de rótulos.

Amostragem de participantes

Quando apenas uma fração dos clientes é selecionada a cada rodada, o SCAFFOLD não funciona de maneira eficaz, e os outros algoritmos apresentam uma acurácia muito instável durante o treinamento. A instabilidade ocorre porque a distribuição de dados entre as rodadas pode variar muito devido à amostragem. O SCAFFOLD falha porque suas

variáveis de controle não são atualizadas com frequência, levando a uma estimativa imprecisa da direção de atualização.

Escalabilidade

A acurácia de todos os algoritmos diminui significativamente à medida que o número de participantes aumenta. Com mais participantes, a quantidade de dados por participantes é menor, facilitando o overfitting durante o treinamento local.

Eficiência

O FedProx tem um custo computacional muito maior em comparação com os outros, pois modifica a loss function, adicionando sobrecarga a cada passo do cálculo do gradiente local. O SCAFFOLD tem um custo de comunicação duas vezes maior que o dos outros, pois precisa transmitir as variáveis de controle (adicionais) em cada rodada.

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: **9 de out. de 2025**

Participantes da Entrega **[matriculados em Residência em IA]**:

MARCELO HENRIQUE ALVES PEREIRA SOBRINHO

Entrega: **[descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]**

Nas Semanas anteriores da Residência em IA, eu havia escolhido o tema **Aprendizado Federado,** dentro da intersecção das grandes áreas de **Computação Distribuída** e **Inteligência Artificial.** Eu havia feito a leitura do artigo seminal sobre esse tema e também busquei informações sobre os maiores desafios dessa área. O intuito era entender melhor as fundamentações teóricas para, nessa última Semana, partir para a prática.

Framework - NVIDIA FLARE

Para a sexta Semana da Residência em IA, iniciei meus estudos explorando a documentação oficial do NVIDIA FLARE, um framework voltado para Aprendizado Federado que possibilita a simulação e execução de experimentos distribuídos em diferentes cenários. Como parte prática, realizei um *fork* do repositório oficial no GitHub com o objetivo de adaptar e simplificar a execução da aplicação. Essa decisão se deu porque o repositório original contém uma grande quantidade de códigos de exemplo, o que facilitou bastante a reprodução de experimentos básicos.

Github-NVIDIAFLARE-ResIA

Nesse repositório, eu redigi um texto explicando o que é o NVIDIA FLARE de maneira holística e também introduzi um tutorial de como usar o ambiente simulado desse *framework* (está no README).

Consegui testar o ambiente simulado na minha própria máquina com a seguinte configuração:

- Dataset: CIFAR10
- Número de usuários: 10
- Batch size de cada usuário: 16
- Número de rodadas de comunicação: 2
- Número de épocas para cada usuário: 2

O Dataset foi misturado de maneira aleatória e enviado para cada cliente (ainda não estou criando simulações com dados não-IID). O teste ocorreu com sucesso e agora eu tenho a capacidade de conseguir configurar o ambiente simulado para experimentos mais difíceis.

As limitações do simulador correspondem às limitações da própria máquina que atua como servidor. Eu havia tentado rodar a simulação com um número muito alto de clientes, mas não obtive sucesso (mesmo utilizando uma RTX 3090

de 24 GB). No entanto, para o meu caso de uso, que consiste em replicar experimentos de artigos científicos, normalmente não é utilizado um número elevado de clientes.

O NVIDIA FLARE também conta com ferramentas de deploy para aplicações do mundo real, porém ainda não explorei.

Artigos mais recentes

Realizei a busca por artigos mais recentes a respeito da questão dos algoritmos de Aprendizado Federado. A ideia era encontrar algum artigo que unisse os pontos positivos dos algoritmos federados que eu havia estudado anteriormente. Com isso, achei o seguinte artigo mais relevante: **FedHybrid: Unifying Aggregation Strategies to Optimize Federated Learning on Non-IID Dataset (2025).**

Revisão de probabilidade e estatística

Por fim, revisei o Capítulo 3 do livro *Noções de Probabilidade e Estatística*, com o objetivo de reforçar as notações matemáticas que são bastante utilizadas por pesquisadores na formulação e descrição dos algoritmos de Aprendizado Federado.

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Para o próximo Gate, tenho os seguintes objetivos:

- Fazer a leitura do artigo **FedHybrid: Unifying Aggregation Strategies to Optimize Federated Learning on Non-IID Dataset,** anotar e entender os resultados que os pesquisadores obtiveram;
- Explorar mais o NVIDIA FLARE, principalmente a parte de como os datasets são distribuídos entre os usuários.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

APÊNDICE 4

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: **16 de out. de 2025**

Participantes da Entrega **[matriculados em Residência em IA]**:

MARCELO HENRIQUE ALVES PEREIRA SOBRINHO

Entrega: **[descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]**

Nas Semanas anteriores, escolhi o **Aprendizado Federado** como tema central da minha Residência em Inteligência Artificial. Inicialmente, realizei a leitura do artigo seminal sobre o tema para fundamentar meu estudo. Em seguida, coletei diversos artigos que abordam o principal desafio dessa área: os dados não-IID (não independentes e não identicamente distribuídos). Esses estudos apresentam experimentos que simulam cenários reais de heterogeneidade dos dados entre clientes e realizam comparações detalhadas entre diferentes algoritmos federados, destacando suas vantagens e limitações nesse contexto.

Nesta última Semana, eu tinha planejado estudar como o **NVIDIA-FLARE** fazia o particionamento dos dados entre os usuários. Porém, percebi logo de início que o particionamento era papel do programador, e que os exemplos de simulação que tinham no repositório, apenas passava o dataset completo para cada usuário. Portanto, eu criei o meu próprio método de particionamento baseado na **distribuição de Dirichlet**, que é uma das formas mais utilizadas para simular dados não-IID e que muitos pesquisadores utilizam.

Implementei um sistema de divisão não-IID do dataset CIFAR-10 utilizando a distribuição de Dirichlet com α = 0.3, que simula heterogeneidade moderada entre clientes. As partições são armazenadas em cache (arquivos .npy) para garantir reprodutibilidade dos experimentos.

https://github.com/marcelinhohaps/NVFlare-ResIA/blob/main/custom/data_loader.py

Além disso, nesta pasta também consta os arquivos do lado do cliente e do lado do servidor criados por mim, e que estão integrados com o DataLoader personalizado. **É altamente recomendado fazer a leitura do arquivo "README.md" que está na pasta "custom" do repositório!**

https://github.com/marcelinhohaps/NVFlare-ResIA/tree/main/custom

Paralelamente, fiz a leitura do artigo **"FedHybrid: Unifying Aggregation Strategies to Optimize Federated Learning on Non-IID Dataset".** Os pesquisadores testaram um novo algoritmo de aprendizado federado, que combina as vantagens de três existentes: **FedAvg**, **FedProx** e **SCAFFOLD**.

Resumo - FedHybrid: Unifying Aggregation Strategies to Optimize Federated Learning on Non-IID Da…

Por fim, decidi escolher o artigo **"Federated Learning on Non-IID Data Silos: An Experimental Study"**

para realizar a replicação dos experimentos. No entanto, pretendo incluir um diferencial: **Investigar o impacto do uso de redes neurais mais robustas,** considerando o limite prático imposto pela capacidade computacional dos dispositivos na borda, já que modelos muito complexos podem não ser viáveis nesses dispositivos.

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

A minha ideia, a partir de agora, é reproduzir o artigo científico mencionado anteriormente com o adicional de avaliar o custo computacional de modelos mais robustos. Portanto, para a próxima Semana, eu gostaria de:

- Montar um documento explicando detalhadamente a metodologia utilizada pelos pesquisadores.
- Continuar explorando a documentação do NVIDIA-FLARE para ter maior base caso eu precise fazer algumas alterações em outros módulos.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Resumo - FedHybrid: Unifying Aggregation Strategies to Optimize Federated Learning on Non-IID Dataset. Documento citado no Termo de Aceite de Entrega do dia 16 de outubro de 2025

Resumo - FedHybrid: Unifying Aggregation Strategies to Optimize Federated Learning on Non-IID Dataset

O artigo apresenta o FedHybrid, um algoritmo que tem o objetivo de unificar diferentes estratégias de agregação no aprendizado federado para otimizar o treinamento em cenários com dados não-IID. A dificuldade principal nesses cenários é que os dados distribuídos pelos clientes são heterogêneos, o que pode levar a uma convergência lenta e menor qualidade dos modelos treinados.

O FedHybrid combina três técnicas já conhecidas: FedAvg, FedProx e FedScaffold. O FedAvg utiliza a média simples dos modelos locais, o que é eficiente mas pode falhar com dados heterogêneos. O FedProx adiciona um termo proximal ao objetivo de otimização, controlando a distância do modelo local ao global, ajudando a estabilizar o treinamento. Já o FedScaffold introduz variáveis de controle para reduzir a variância das atualizações dos clientes, melhorando a precisão e velocidade da convergência. O FedHybrid integra esses três componentes, aproveitando os pontos fortes de cada um.

Para validar sua abordagem, os autores testaram o FedHybrid em dois datasets padrão de visão computacional, MNIST (imagens de dígitos) e CIFAR-10 (imagens em 10 classes). Foram simulados 100 clientes, cada um com dados particionados para representar a heterogeneidade típica do mundo real, e realizados mais de 100 ciclos de comunicação entre clientes e servidor. O FedHybrid apresentou uma acurácia superior, alcançando 94,12% no MNIST e 93,52% no CIFAR-10, superando consistentemente FedAvg, FedProx e FedScaffold isoladamente. Além disso, mostrou convergência mais rápida, economizando tempo de treinamento.

O artigo ainda aborda questões práticas, como os impactos do tamanho do lote, taxa de aprendizado, e o papel do parâmetro proximal no desempenho. Também reconhece limitações, como a escala do experimento e o uso de datasets relativamente pequenos, sugerindo como trabalho futuro a aplicação da técnica em

problemas maiores e mais diversos, além de explorar a integração com outras técnicas de privacidade e segurança.

Em resumo, FedHybrid representa um avanço para o aprendizado federado em ambientes reais, nos quais a heterogeneidade dos dados e sistemas impõe desafios significativos para a eficiência e qualidade dos modelos, oferecendo um método mais robusto e eficiente para lidar com esses problemas.

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: **22 de out. de 2025**

Participantes da Entrega **[matriculados em Residência em IA]**:

MARCELO HENRIQUE ALVES PEREIRA SOBRINHO

Entrega: **[descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]**

Nas primeiras semanas da Residência em IA, escolhi inicialmente estudar Computação Distribuída. Ao aprofundar a intersecção entre Computação Distribuída e Machine Learning, o tema de Aprendizado Federado despertou grande interesse. Busquei os fundamentos, os algoritmos e li diversos artigos relacionados. Posteriormente, decidi replicar um artigo relevante da área: **"Federated Learning on Non-IID Data Silos: An Experimental Study".**

Esse artigo central aborda o desafio crucial da heterogeneidade dos dados em aprendizado federado, especialmente quando os dados dos clientes não são independentes e identicamente distribuídos (non-IID). Os autores propõem estratégias abrangentes de particionamento de dados para simular diferentes tipos típicos de distribuição não-IID, **incluindo desbalanceamento na distribuição dos rótulos**, **das características** e na **quantidade de dados entre clientes**. Com uma extensa bateria de experimentos, eles avaliaram o desempenho de algoritmos federados e evidenciam que nenhum método domina todos os cenários.

Nesta última Semana, concentrei meus esforços na organização da seção prática. Continuei escrevendo no documento de resumo que havia elaborado para o artigo. A partir da página 8, está o detalhamento dos experimentos.

Resumo - Federated Learning on Non-IID Data Silos: An Experimental Study

Encontrei algumas inconsistências entre as características dos datasets descritos no artigo e os datasets que eu baixei da fonte oficial (número diferente de features etc). Por isso, eu tive que criar um arquivo .ipynb para verificar cada detalhe dos datasets para montar a tabela que está no documento e evitar divergência de informações.

https://github.com/marcelinhohaps/NVFlare-ResIA/blob/main/custom/utils/datasets_experiment.ipynb

Também realizei um commit para fazer a organização do repositório e adicionar os modelos que foram utilizados no experimento

https://github.com/NVIDIA/NVFlare/commit/f4c2e017917efab6188841af5cc2f233d59116a8

Portanto, para o experimento, eu já realizei:

- Identificação, coleta e detalhamento dos conjuntos de dados
- Detalhamento do experimento de maneira holística
- Criação dos métodos (funções) de particionamento
- Criação dos modelos utilizados

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Continuar a execução do experimento do artigo. Eu preciso, para a próxima Semana:

- Realizar o experimento para o primeiro caso (**desvio de quantidade de dados entre clientes**) e registrar os resultados.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Federated Learning on Non-IID Data Silos: An Experimental Study - Seção de resultado dos experimentos - Seção de detalhamento dos experimentos. Documento citado no Termo de Aceite de Entrega do dia 22 de outubro de 2025

Experimentos

Datasets de Imagens

- **MNIST**: composto por imagens em escala de cinza de dígitos manuscritos (0 a 9), com tamanho fixo de 28x28 pixels. Dividido em 60.000 imagens para treino e 10.000 para teste, ele é amplamente utilizado para tarefas de classificação de dígitos e serve como benchmark para modelos iniciais de visão computacional.
- **FMNIST**: Inspirado no MNIST, o FMNIST apresenta imagens em escala de cinza, com 28x28 pixels, porém as imagens são de categorias de roupas e acessórios (como camisetas, calças, sapatilhas). Com 60.000 exemplos para treino e 10.000 para teste, FMNIST é empregado como alternativa mais realista e desafiadora ao MNIST clássico para tarefas de classificação de imagens.
- **CIFAR-10:** O CIFAR-10 contém imagens coloridas (RGB) de pequena resolução (32x32 pixels) divididas em 10 classes distintas, como animais (cães, gatos, pássaros) e veículos (aviões, carros, barcos). Seu desafio inclui a diversidade de classes e a complexidade visual comparada a datasets como MNIST.
- **SVHN:** O SVHN é uma coleção de mais de imagens coloridas de números retirados de fotos de casas do Google Street View. Cada imagem contém um dígito em diferentes contextos, iluminação e ângulos, tornando o dataset útil para testes de reconhecimento em imagens do “mundo real”. As imagens são em RGB, com resolução variável. SVHN exemplifica desafios práticos de OCR (reconhecimento óptico de caracteres) em ambientes não controlados.

Datasets Tabulares

- **Adult (Census Income):** O Adult dataset reúne informações demográficas e socioeconômicas de aproximadamente 48.000 indivíduos nos EUA, com o

objetivo de prever se a renda anual ultrapassa 50 mil dólares. Contém atributos categóricos (sexo, estado civil, raça, ocupação) e numéricos (idade, horas trabalhadas por semana). É muito usado para problemas de classificação binária.

- **Covtype:** O Covertype inclui dados ambientais coletados em áreas florestais das Montanhas Rochosas, contendo cerca de 580.000 registros com atributos que descrevem variáveis climáticas, topográficas e do solo. O objetivo é classificar o tipo de cobertura do solo (7 classes).

Datasets	Instâncias de treinamento	Instâncias de teste	Features	Classes
CIFAR-10	50000	10000	1024	10
MNIST	60000	10000	784	10
FMNIST	60000	10000	784	10
SVHN	73257	26032	1024	10
adult	32561	16281	100	2
rcv1	15182	5060	47236	103
covtype	435759	145253	54	7

Tabela 1: Características dos datasets utilizados no experimento

Métodos de Particionamento

- **Quantity skew:** Todos os clientes podem ter a mesma distribuição conjunta (features e classes), mas os tamanhos dos conjuntos de dados são diferentes, gerando uma desproporção na quantidade de amostras para treinamento local. Foi implementado da seguinte forma:
    - 1 - Utilizando a distribuição de Dirichlet para alocar diferentes proporções de dados a cada cliente controlando o desequilíbrio na quantidade de amostras.

Modelos:

- **Multi Layer Perceptron (MLP)**: Para os dados tabulares, foi utilizada uma MLP com três camadas ocultas, contendo 32, 16 e 8 unidades, respectivamente;
- **Convolutional Neural Network (CNN)**: Para os dados de imagens, foi utilizada uma CNN composta por duas camadas de convolução 5x5 seguidas por camadas de max pooling 2x2; a primeira com 6 canais e a segunda com 16 canais; além de duas camadas totalmente conectadas com a função de ativação ReLU (a primeira com 120 unidades e a segunda com 84 unidades).

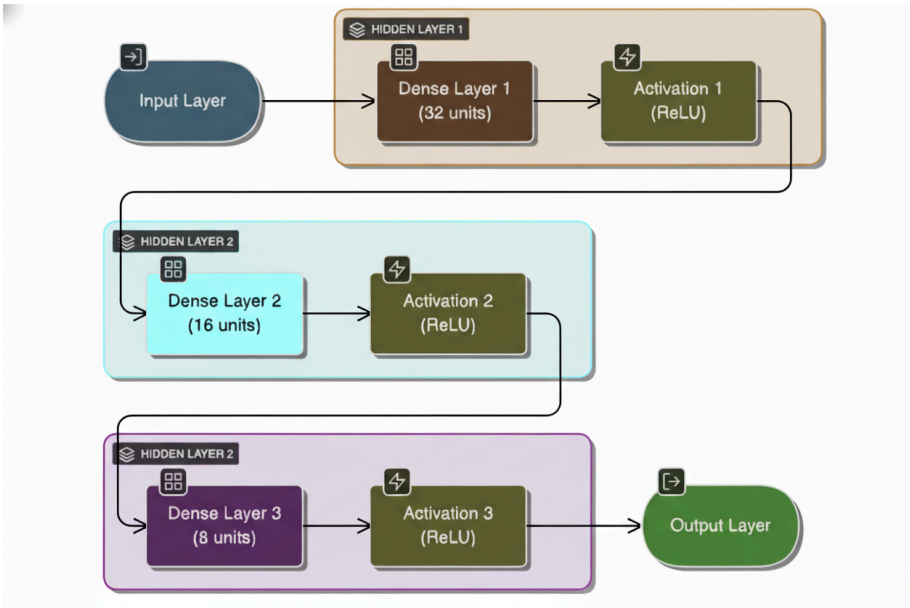

Figura 1: Arquitetura da MLP utilizada nos experimentos.

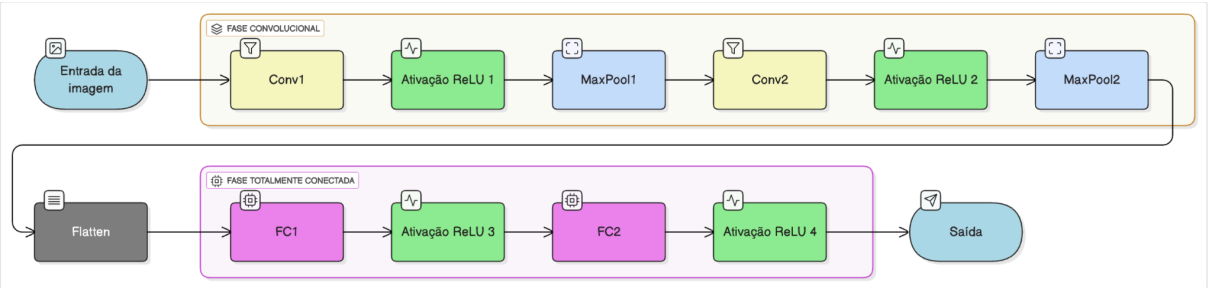

Figura 2: Arquitetura da CNN utilizada nos experimentos.

Hiperparâmetros e Métrica de Avaliação

Hiperparâmetros do Treinamento Local:

- Otimizador: SGD
- Batch size: 64
- Número de épocas: 10
- Taxa de Aprendizado: 0,1 (apenas para o dataset RCV1) e 0,01 para os outros
- Momentum: 0,9

Hiperparâmetros do Aprendizado Federado:

- Número de clientes: 10
- Número de rodadas de comunicação: 50

Métrica de Avaliação:

A métrica utilizada foi a acurácia top-1.

Infraestrutura utilizada:

- RTX 3090 24GB
- 32GB RAM

APÊNDICE 5

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: **6 de nov. de 2025**

Participantes da Entrega **[matriculados em Residência em IA]**:

MARCELO HENRIQUE ALVES PEREIRA SOBRINHO

Entrega: **[descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]**

Nas primeiras Semanas da Residência em IA, decidi me aprofundar nos temas de **Computação Distribuída** e **Machine Learning**. Com o avanço das Semanas, optei por direcionar meus estudos para o **Aprendizado Federado**, uma área que se encontra na intersecção entre esses dois campos. Nesse período, realizei a leitura do artigo seminal que introduz o **FedAvg**, o principal algoritmo responsável por viabilizar o aprendizado federado.

Além disso, busquei trabalhos que abordassem o desafio dos **dados não-iid**, encontrando o artigo *"Federated Learning on Non-IID Data Silos: An Experimental Study"*, no qual decidi replicar a metodologia parcialmente. Para a implementação dos experimentos, utilizei o **framework NVIDIA-FLARE**, que explorei e pratiquei ao longo das Semanas.

Nesta última Semana, conforme planejado no gate anterior, realizei um primeiro experimento envolvendo **desvio de dados entre usuários com diferentes quantidades de amostras**, investigando como essa variação impacta o desempenho do modelo federado.

link da pasta no repositório: https://github.com/marcelinhohaps/NVFlare-ResIA/tree/main/custom

A métrica avaliada foi a de **Acurácia.**

Obtive os seguintes resultados com **10 usuários e 50 rodadas de comunicação**:

- FedAvg:
  - **adult**: 86,4% -> **Artigo**: 82,2%
  - **cifar10**: 58% -> **Artigo**: 72%
  - **covtype**: 85,3% -> **Artigo**: 88,1%
  - **fmnist**: 89,6% -> **Artigo**: 89,4%
  - **mnist**: 98,9% -> **Artigo**: 99,2%
  - **svhn**: 81,4% -> **Artigo**: 88,3%
- FedProx:
  - **adult**: 86,3% -> **Artigo**: 84,8%
  - **cifar10**: 55,8% -> **Artigo**: 71,2%
  - **covtype**: 82,8% -> **Artigo**: 84,6%

- **fmnist**: 89,9% -> **Artigo**: 89,7%
- **mnist**: 99,3% -> **Artigo**: 99,2%
- **svhn**: 81,9% -> **Artigo**: 88,4%

Eu também consegui criar uma forma de monitorar todo o processo de treinamento no **Weights and Bias** (WandB), mas não consegui compartilhar um link público. Eu precisaria criar um report para cada projeto e disponibilizar o link.

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

- Organizar os resultados dos experimentos de forma mais visual e apresentá-los lado a lado com o artigo para facilitar a análise comparativa.
- Criar reports no WandB
- Iniciar a fase 2 dos experimentos, onde vou iniciar o processamento com o outro cenário de desvio de dados: **desvio na distribuição de features**

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

APÊNDICE 6

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: **12 de nov. de 2025**

Participantes da Entrega **[matriculados em Residência em IA]:**

MARCELO HENRIQUE ALVES PEREIRA SOBRINHO

Entrega: **[descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]**

Tema: **Aprendizado Federado**

Revisão

Nas primeiras Semanas, optei por iniciar meus estudos nas áreas de **Computação Distribuída** e **Aprendizado de Máquina**. Durante esse período, li diversos artigos sobre MapReduce e Spark, além de outros materiais relacionados a esses temas. Ao explorar um survey que abordava a intersecção entre Computação Distribuída e Machine Learning, conheci o conceito de Aprendizado Federado. A partir disso, decidi me aprofundar nesse assunto e torná-lo o foco principal do meu estudo durante a Residência.

Aprofundei meus estudos nos principais algoritmos de Aprendizado Federado, buscando compreender tanto seus fundamentos teóricos quanto sua aplicação prática. Analisei como esses algoritmos realizam a agregação de modelos locais, a comunicação entre cliente e servidor, e as estratégias utilizadas para lidar com dados heterogêneos (non-IID).

Entre os algoritmos estudados, destaquei o **FedAvg**, por ser o mais clássico e amplamente utilizado, além de explorar variações como **FedProx**, **FedNova** e **SCAFFOLD**, que propõem melhorias em estabilidade, desempenho e convergência. Também procurei entender os desafios envolvidos nesse tipo de aprendizado, como o balanceamento entre eficiência de comunicação, privacidade e desempenho global do modelo.

Além disso, li o artigo ***"Federated Learning on Non-IID Data Silos: An Experimental Study"***, no qual os pesquisadores realizaram um benchmark de quatro algoritmos de Aprendizado Federado testados em diversos *datasets*. Decidi replicar parcialmente esse experimento e obtive resultados bastante semelhantes aos apresentados pelos autores.

Optei por utilizar o framework **NVIDIA-FLARE**, estudando suas principais componentes e empregando o modo de simulação para conduzir os experimentos descritos no artigo.

Os experimentos estão descritos aqui:

Resumo - Federated Learning on Non-IID Data Silos: An Experimental Study

Com exceção de um *dataset* (**CIFAR-10**), a acurácia obtida nos meus experimentos apresentou uma diferença de cerca de **20%** em relação aos resultados relatados no artigo. Diante disso, realizei uma investigação detalhada: refiz os testes, revisei o código e voltei à leitura do artigo para verificar se alguma informação importante havia passado despercebida. Ainda assim, não consegui identificar a causa exata dessa discrepância nos resultados.

Gate 10

Nesta última Semana, decidi colocar as métricas de treinamento no documento do experimento do artigo:
Resumo - Federated Learning on Non-IID Data Silos: An Experimental Study

Além disso, senti a necessidade de colocar o Aprendizado Federado em prática. Percebi que, ao utilizar o modo de simulação, eu estava lidando principalmente com Computação Paralela, e não de fato com Computação Distribuída. Por isso, iniciei meus estudos sobre os modos de **desenvolvimento** e **produção** do **NVIDIA-FLARE**. Essa etapa representou um grande desafio, já que quase não existiam tutoriais didáticos disponíveis para orientar o processo. Dessa forma, precisei me apoiar diretamente na documentação oficial e avançar por meio de testes e experimentações.

No final, eu consegui realizar uma rodada de comunicação completa de maneira condizente com a teoria do Aprendizado Federado. Então, eu consegui configurar as máquinas, configurar a comunicação entre elas, fazer o provisionamento, distribuir os Startup Kits e submeter um job para o único cliente. Todo o processo foi documentado aqui em uma forma de tutorial também:
https://www.overleaf.com/read/mngvfgtfrhkd#3192a7

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Consolidar tudo o que foi estudado e deixar mais organizado.

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

O documento desta Semana ainda não está formatado nos moldes dos documentos da Residência. Vou providenciar isso. Fico feliz de ter conseguido chegar até essa parte de provisionar um sistema de Aprendizado Federado. Até então, isso parecia impossível (eu até tinha falado isso no começo).

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

Federated Learning on Non-IID Data Silos: An Experimental Study - Seção de resultado dos experimentos. Documento citado no Termo de Aceite de Entrega do dia 6 de novembro de 2025

Resultados do experimento

Os experimentos foram conduzidos utilizando os hiperparâmetros descritos anteriormente. A Tabela 2 apresenta os valores de acurácia obtidos para cada combinação de algoritmo e conjunto de dados. O framework empregado foi o NVIDIA FLARE, executado no modo de simulação sobre a infraestrutura descrita previamente. Cada rodada de comunicação teve duração aproximada de um minuto, variando conforme o dataset utilizado.

O repositório do experimento é o https://github.com/marcelinhohaps/NVFlare-ResIA/tree/main/custom/2102.02079_experiment

Dataset	FedAvg	FedProx
CIFAR10	58%	55,8%
MNIST	98,9%	99,3%
FMNIST	89,6%	89,9%
covtype	85,3%	82,8%
svhn	81,4%	81,9%
adult	86,4%	86,3%

Tabela 2: Resultados do experimento com o desvio na quantidade de amostras para cada cliente.

Figura 3: Métricas de treinamento para o dataset SVHN com o algoritmo FedProx.

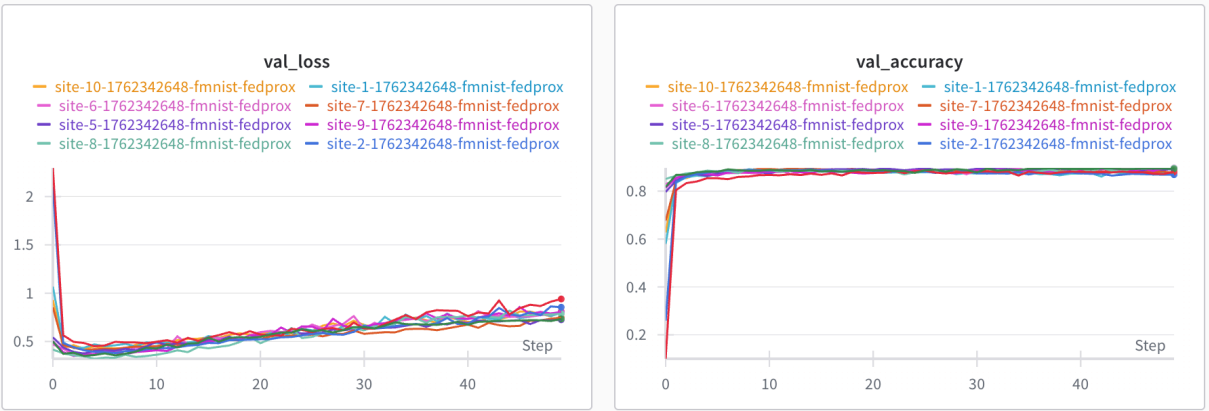

Figura 4: Métricas de treinamento para o dataset FMNIST com o algoritmo FedProx.

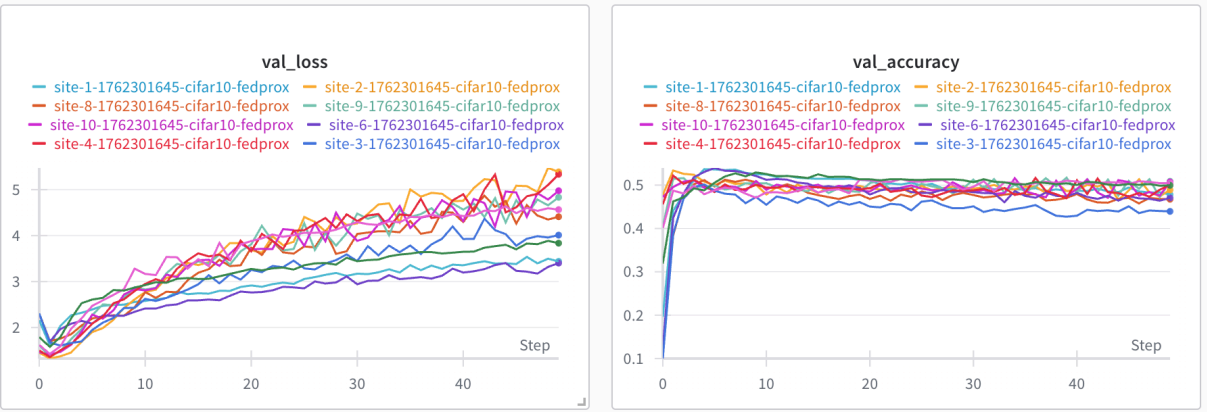

Figure 17.

Figura 5: Métricas de treinamento para o dataset CIFAR-10 com o algoritmo FedProx.

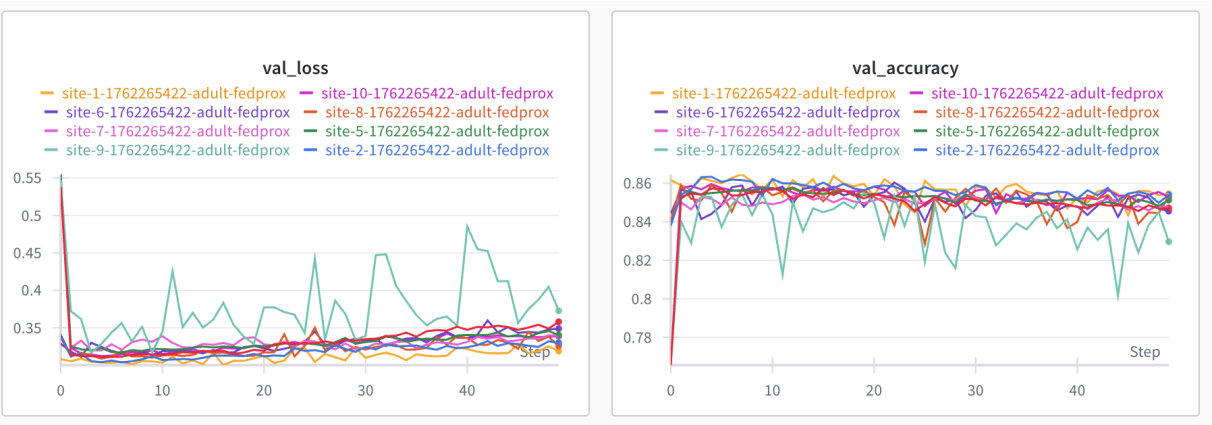

Figure 18.

Figura 6: Métricas de treinamento para o dataset Adult com o algoritmo FedProx.

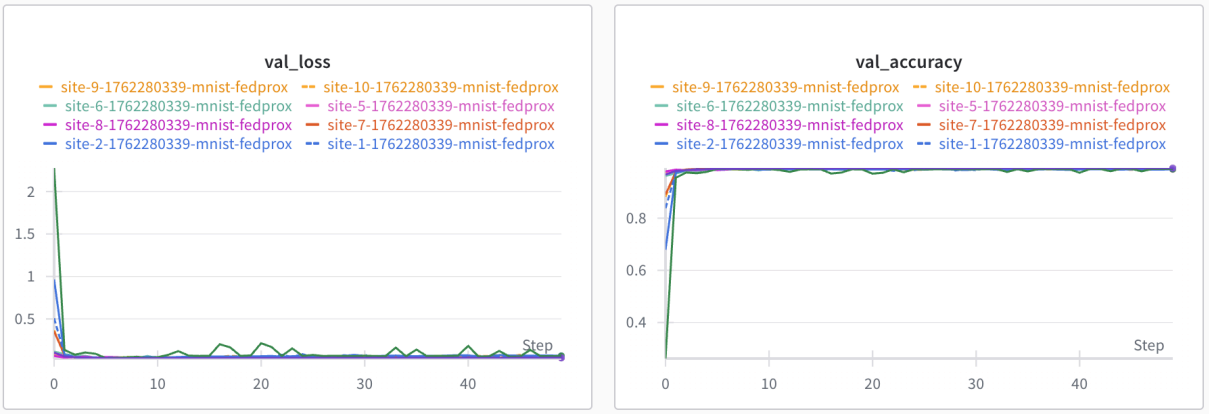

Figura 7: Métricas de treinamento para o dataset MNIST com o algoritmo FedProx.

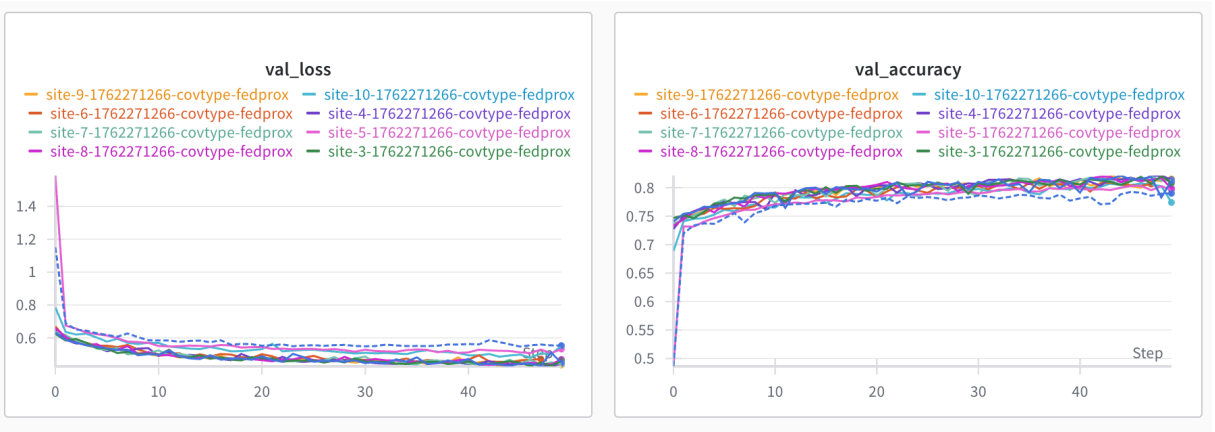

Figura 8: Métricas de treinamento para o dataset Covtype com o algoritmo FedProx.

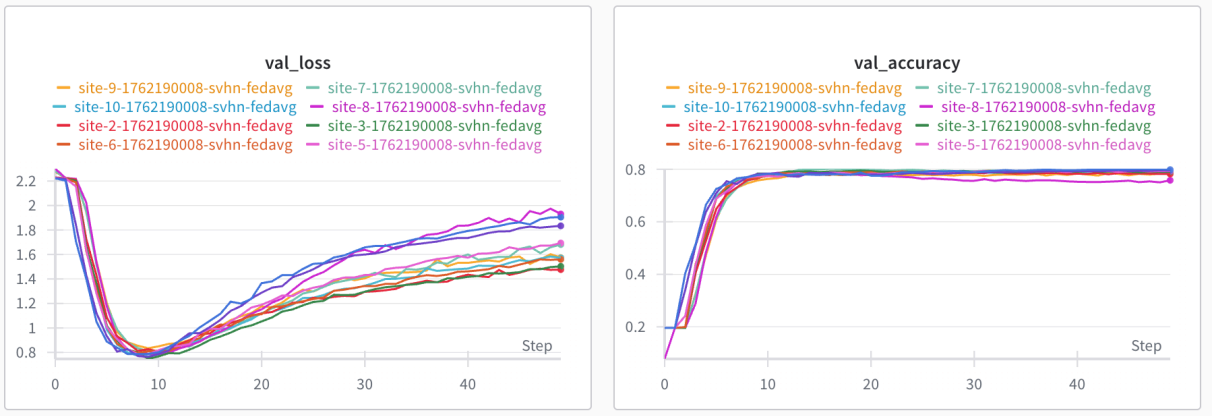

Figura 9: Métricas de treinamento para o dataset SVHN com o algoritmo FedAvg.

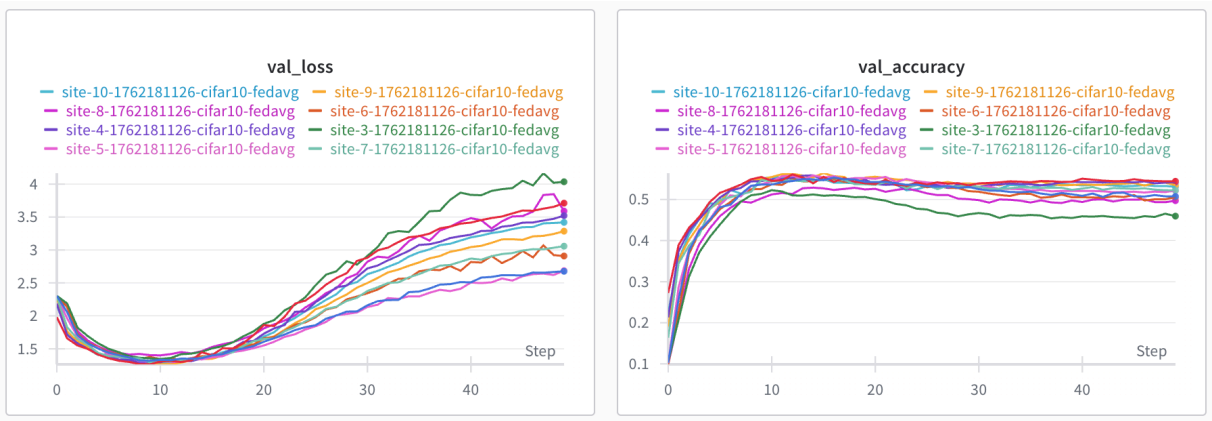

Figure 22.

Figura 10: Métricas de treinamento para o dataset CIFAR-10 com o algoritmo FedAvg.

Figura 11: Métricas de treinamento para o dataset Adult com o algoritmo FedAvg.

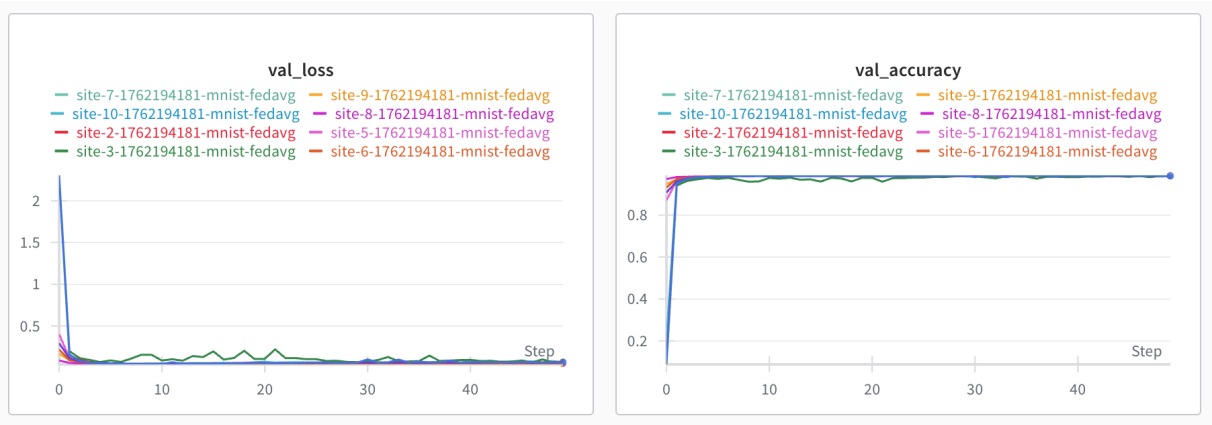

Figura 12: Métricas de treinamento para o dataset MNIST com o algoritmo FedAvg.

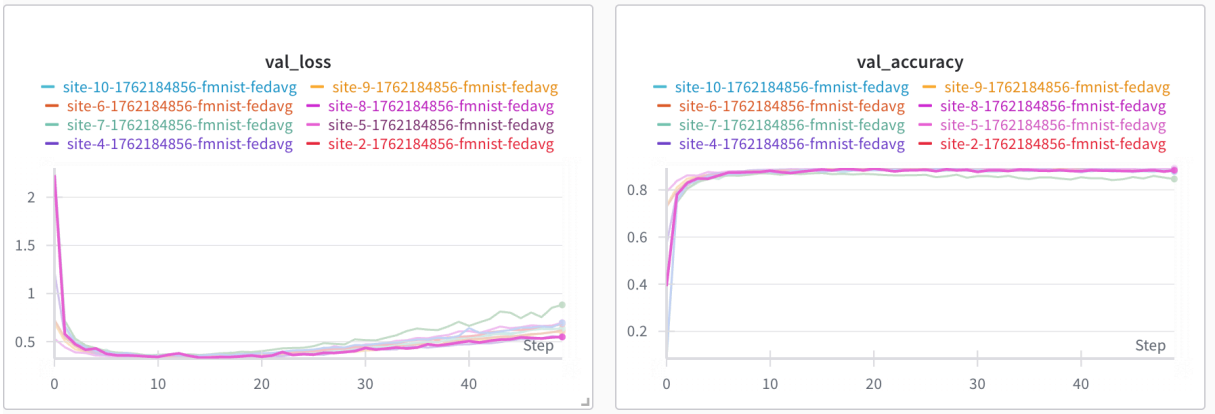

Figura 13: Métricas de treinamento para o dataset FMNIST com o algoritmo FedAvg.

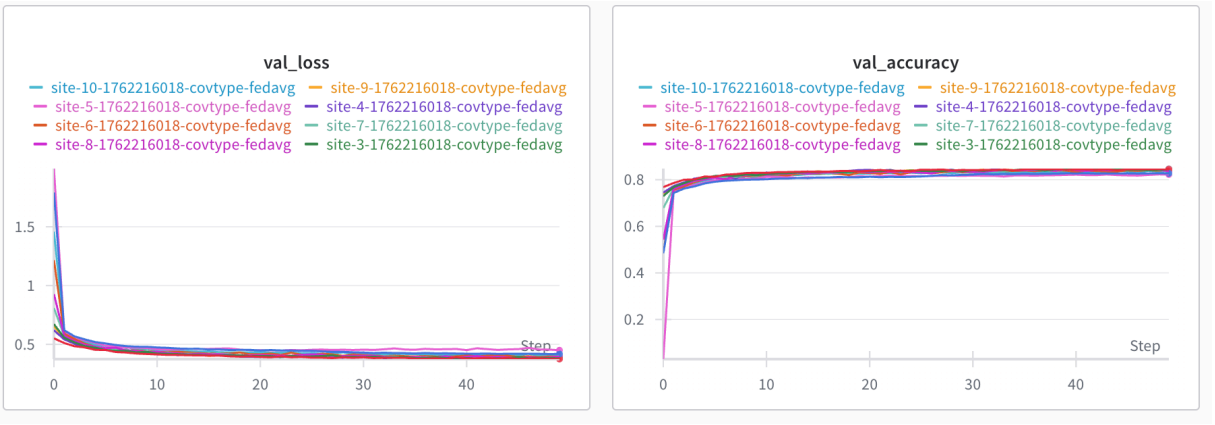

Figura 14: Métricas de treinamento para o dataset Covtype com o algoritmo FedAvg.

Documento sobre todas as informações sobre Aprendizado Federado e tutorial sobre como provisionar um sistema de treinamento distribuído com o NVIDIA FLARE. Documento citado no Termo de Aceite de Entrega do dia 12 de novembro de 2025

O link disponibilizado abaixo serve como um roteiro técnico para a adoção do Aprendizado Federado. Inicialmente, estabelecem-se as bases teóricas sobre a privacidade de dados (LGPD) e a topologia de rede necessária para o treinamento colaborativo. São descritos os mecanismos de agregação de modelos, incluindo FedAvg e FedProx, como soluções para a heterogeneidade estatística. A aplicação prática é demonstrada através do NVIDIA FLARE, cobrindo o ciclo completo de desenvolvimento: da simulação em ambiente local ao provisionamento de produção. Adicionalmente, o documento especifica a configuração de canais seguros via mTLS e o gerenciamento operacional do sistema através de interfaces visuais e linhas de comando.

https://github.com/marcelinhohaps/NVFlare-ResIA/blob/main/custom/tudo_de_aprendizado_federado.pdf