UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

LUCAS CHAGAS MOREIRA

EXPLORANDO TÉCNICAS DE APRENDIZADO DE MÁQUINA NA ANÁLISE DE MOTORES ELÉTRICOS

Ouro Preto,
2026

LUCAS CHAGAS MOREIRA

EXPLORANDO TÉCNICAS DE APRENDIZADO DE MÁQUINA NA ANÁLISE DE MOTORES ELÉTRICOS

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Ouro Preto,
2026

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

M838e Moreira, Lucas Chagas.
Explorando técnicas de aprendizado de máquina na análise de motores elétricos. [manuscrito] / Lucas Chagas Moreira. - 2026.
51 f.

Orientador: Prof. Dr. Rodrigo César Pedrosa Silva.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Biológicas. Graduação em Ciência da Computação .

1. Redes neurais (Computação). 2. Aprendizado do computador. 3. Arquitetura de rede de computador. 4. Motores elétricos. I. Silva, Rodrigo César Pedrosa. II. Universidade Federal de Ouro Preto. III. Título.

CDU 004.032.26:621.3

Bibliotecário(a) Responsável: Renata Mara de Almeida - CRB-7: 6328

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E BIOLOGICAS
DEPARTAMENTO DE COMPUTACAO

FOLHA DE APROVAÇÃO

Lucas Chagas Moreira

Explorando Técnicas de Aprendizado de Máquina na Análise de Motores Elétricos

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Ciência da Computação

Aprovada em 27 de Fevereiro de 2026

Membros da banca

Rodrigo Cesar Pedrosa Silva (Orientador) - Doutor - Universidade Federal de Ouro Preto
Gabriel Bicalho Ferreira (Examinador) - Mestre - SMi Engenharia
Vinícius Nascimento Targa (Examinador) - Bacharel - Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Ouro Preto

Rodrigo Cesar Pedrosa Silva, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 27/02/2026

Documento assinado eletronicamente por **Rodrigo Cesar Pedrosa Silva**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 03/03/2026, às 09:27, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **1062476** e o código CRC **52E452C5**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: 3135591692 - www.ufop.br

Agradecimentos

Agradeço, em primeiro lugar, aos meus orientadores, Dr. Rodrigo César Pedrosa Silva e Dr. Pedro Henrique Lopes Silva, pela orientação e pelo incentivo ao longo da minha trajetória acadêmica. Sou grato pela paciência em corrigir meus erros, pelo apoio e pelas valiosas ideias que ampliaram minha visão e contribuíram para o meu desenvolvimento na pesquisa.

Estendo também meus agradecimentos aos colegas que conheci durante a graduação: Felipe Braz, Nicolas Mendes, Matheus Peixoto, Pedro Henrique, Pedro Morais e Gabriel Saldanha, pela amizade e momentos de descontração que tornaram o percurso mais leve, agradável e menos exaustivo.

Resumo

Os motores síncronos de ímãs permanentes internos (IPMSMs) vêm ganhando destaque em aplicações de tração elétrica devido à elevada eficiência energética e à alta densidade de potência. Métodos tradicionais, como a Análise por Elementos Finitos (FEA), embora precisos, apresentam alto custo computacional, o que motivou o uso de técnicas de aprendizado de máquina para construir modelos substitutos capazes de prever parâmetros de desempenho de forma mais ágil. Contudo, a restrição de acesso a dados privados de diferentes empresas limita a diversidade dos conjuntos de treinamento, prejudicando a generalização dos modelos. Nesse contexto, este trabalho tem como objetivo avaliar a viabilidade de técnicas de aprendizado de máquina na modelagem preditiva de parâmetros de desempenho de IPMSMs, com o intuito de criar um método substituto para a análise de IPMSMs. Foram utilizados conjuntos de dados de três topologias distintas de rotores de IPMSM (2D, Nabla e V), avaliando modelos densos, *AutoML*, *transfer learning*, *autoencoders* e aprendizado federado. Os resultados indicam que redes neurais densas e *AutoML* superaram o modelo XGBoost em diversas métricas, enquanto *autoencoders* possibilitaram a criação de modelos gerais para múltiplas bases. Já a abordagem federada obteve desempenho próximo ao da centralizada, com perdas modestas que podem ser reduzidas por meio do ajuste de hiperparâmetros e arquiteturas. Conclui-se que o aprendizado de máquina é uma alternativa viável para a análise de motores elétricos, mantendo a qualidade preditiva e abrindo caminho para pesquisas futuras em estratégias de agregação e generalização.

Palavras-chave: Redes Neurais. Motores Elétricos. *Transferlearning*. Aprendizado Federado. *Autoencoders*

Abstract

Interior permanent magnet synchronous motors (IPMSMs) have gained prominence in electric traction applications due to their high energy efficiency and power density. Traditional methods such as Finite Element Analysis (FEA), although accurate, entail high computational cost, which has motivated the use of machine learning techniques to build surrogate models capable of predicting performance parameters more efficiently. However, restricted access to private data from different companies limits the diversity of training sets, harming model generalization. In this context, this work aims to evaluate the feasibility of machine learning techniques for the predictive modeling of IPMSM performance parameters, with the goal of creating a surrogate method for IPMSM analysis. Datasets from three distinct IPMSM rotor topologies (2D, Nabla, and V) were used to evaluate dense neural networks, *AutoML*, *transfer learning*, *autoencoders*, and federated learning. The results indicate that dense neural networks and *AutoML* outperformed the XGBoost model across several metrics, while *autoencoders* enabled the development of general models applicable to multiple datasets. The federated approach achieved performance close to centralized training, with modest losses that can be further reduced through hyperparameter tuning and architectural adjustments. In conclusion, machine learning is a viable alternative for electric motor analysis, maintaining predictive quality and opening opportunities for future research on aggregation strategies and generalization.

Keywords: Neural Networks. Federated Learning. Electric Motors. Transfer Learning. Autoencoders.

Lista de Figuras

Lista de Tabelas

Lista de Códigos

Lista de Abreviaturas e Siglas

ABNT	Associação Brasileira de Normas Técnicas
DECOM	Departamento de Computação
UFOP	Universidade Federal de Ouro Preto
IPMSMs	*Interior permanent magnet synchronous motors*
FEA	*Finite Element Analysis*
FL	*Federated Learning*
MSE	*Mean squared error*
MAE	*Mean absolute error*
MAPE	*Mean absolute percentual error*
ReLU	*Rectified linear unit*
ADAM	*Adaptive Moment Estimation*
FedAvg	*Federated Averaging*

Lista de Símbolos

Γ	Letra grega Gama
Λ	Lambda
ζ	Letra grega minúscula zeta
$\in$	Pertence
σ	Sigmoid

Sumário

1 Introdução

Os motores elétricos têm recebido crescente atenção pelo seu papel na redução das emissões de dióxido de carbono (HERNANDEZ et al., 2020). Em especial, os motores síncronos de ímã permanente interno (*Interior Permanent Magnet Synchronous Motors*, IPMSMs) destacam-se em aplicações de tração veicular graças à elevada densidade de potência, à alta eficiência energética e à excelente resposta dinâmica (SHIMIZU, 2023).

A Análise por Elementos Finitos (*Finite Element Analysis*, FEA) é amplamente utilizada para caracterizar o comportamento de IPMSMs, permitindo estimar parâmetros críticos como torque e perdas sob diferentes condições de operação, reduzindo o tempo de prototipagem no desenvolvimento dos motores. Esses resultados servem de base para projetos que otimizam custos e desempenho (SHIMIZU, 2023). No entanto, a FEA exige tempo de processamento significativo, pois as simulações precisam ser repetidas para múltiplos cenários de ensaio e variantes de projeto, como diferentes níveis de torque e velocidade (SHIN et al., 2014).

Aprendizado de máquina é um subcampo da inteligência artificial que estuda algoritmos capazes de ajustar automaticamente seus parâmetros a partir de grandes volumes de dados, sem a necessidade de modelagem explícita das leis físicas (SIMEONE, 2018). Esse campo engloba três paradigmas principais aprendizado supervisionado, não supervisionado e por reforço que se distinguem pela forma de interação com os dados e pelos objetivos de cada técnica (JORDAN; MITCHELL, 2015). Em aplicações de análise de IPMSMs, técnicas de aprendizado de máquina vêm sendo empregadas para desenvolver modelos substitutos (*surrogate models*) que, treinados com um volume menor de resultados de simulações de FEA, conseguem prever com precisão parâmetros como torque, perdas de ferro e distribuições de fluxo em uma fração do tempo exigido pelas análises tradicionais de FEA (SHIMIZU, 2023).

Porém, mesmo com a adoção de modelos substitutos, persiste um desafio significativo relacionado à privacidade dos dados, que limita a capacidade desses modelos de alcançar desempenho robusto em um portfólio diversificado de motores com diferentes configurações. As informações operacionais são frequentemente protegidas pelas empresas detentoras dos projetos, resultando em um volume insuficiente de dados disponíveis para o treinamento eficaz dos algoritmos.

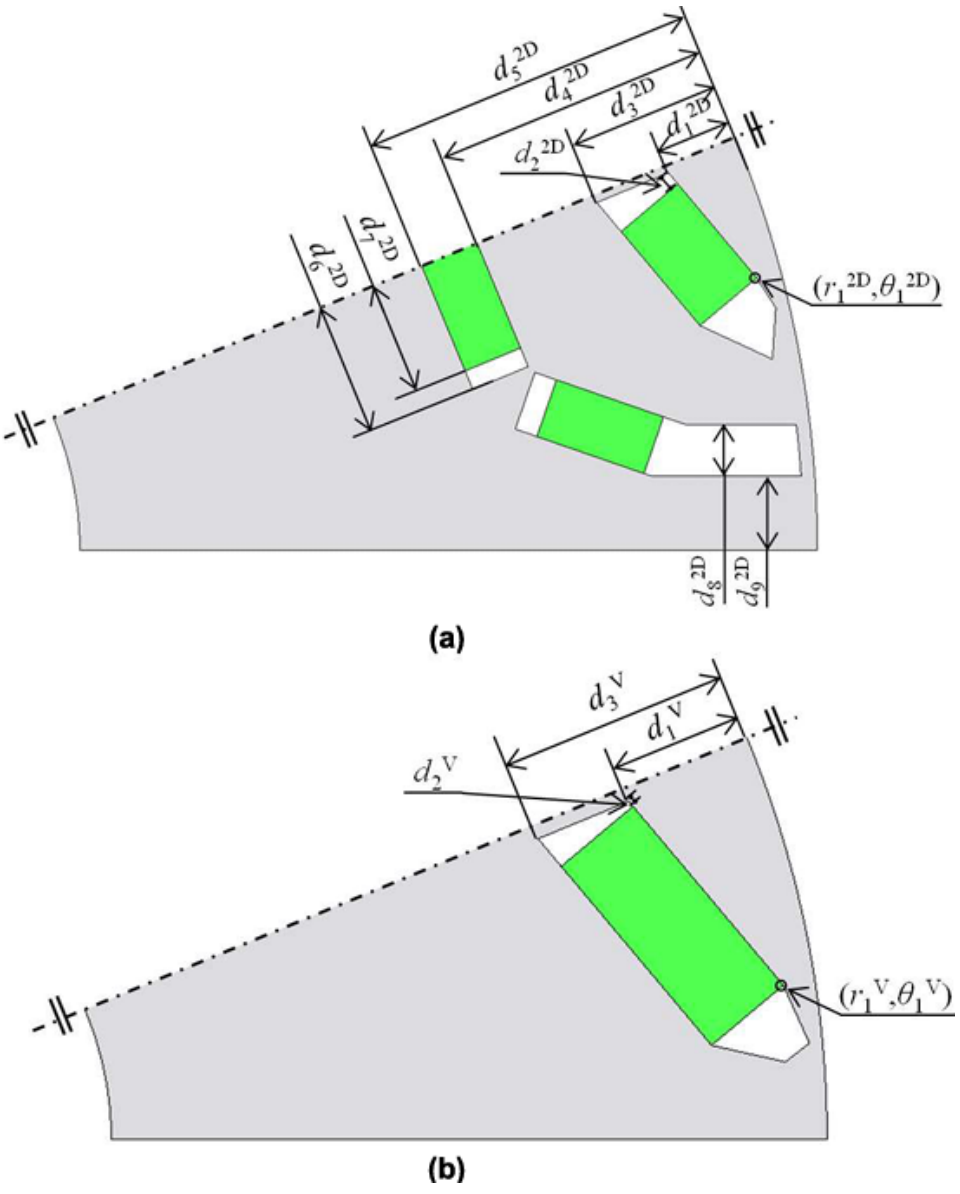
Surgem, então, os métodos de aprendizado federado (*Federated Learning*, FL) como solução ao desafio de privacidade. Trata-se de uma abordagem descentralizada de machine *learning*, na qual um servidor central coordena o processo de treinamento, enquanto os diversos clientes (ou nós) mantêm seus dados localmente. Inicialmente, o servidor distribui um modelo inicial geralmente com parâmetros aleatórios a cada nó participante. Em seguida, cada nó realiza o treinamento localmente usando seu próprio conjunto de dados. Concluída essa etapa em

todos os clientes, apenas os parâmetros atualizados dos modelos são enviados de volta ao servidor, onde ocorre a agregação dessas atualizações, resultando em um modelo global cada vez mais refinado, sem jamais expor os dados individuais de cada nó (ZHANG et al., 2021).

Por conta dessa arquitetura de compartilhar apenas parâmetros atualizados e nunca os dados brutos o aprendizado federado supera as barreiras de privacidade e as exigências burocráticas impostas pelas empresas.

Embora os IPMSMs compartilhem princípios de funcionamento semelhantes, suas geometrias podem variar significativamente. Essa heterogeneidade complica o treinamento por aprendizado federado com dados oriundos de fontes distintas, pois as diferenças estruturais podem impedir que o modelo atinja um nível de convergência aceitável em comparação com o artigo de referencia (SHIMIZU, 2023).

Figura 1.1 – Exemplo de designs de 2 IPMSM (SHIMIZU, 2023).



Fonte: (SHIMIZU, 2023).

Uma técnica amplamente utilizada em aprendizado de máquina para lidar com incompatibilidade de dimensões e heterogeneidade nos dados é o *encoding* (BENGIO; COURVILLE; VINCENT, 2013). *Encoding* consiste em projetar informações originalmente representadas em um espaço de alta (ou diferente) dimensão para outro espaço vetorial de dimensão fixa, preservando ao máximo as características e relações de similaridade do conjunto original. Por meio de uma função de mapeamento frequentemente implementada por um *encoder* neural ou outro modelo otimizado é possível converter variáveis categóricas, numéricas ou mesmo descritores geométricos complexos em vetores densos, mantendo o maior número possível de informações relevantes.

No contexto do projeto de motores elétricos IPMSMs 1.1, cada variação de design altera a geometria e, consequentemente, a forma e a distribuição dos dados gerados. Assim, diferentes configurações tendem a produzir bases com estruturas e dimensões incompatíveis, dificultando o treinamento de modelos únicos e comparáveis. Nesse cenário, o uso de técnicas de *encoding* permite representar de maneira sistemática as variações geométricas e funcionais de cada configuração em um espaço latente compartilhado. Ao transformar descrições heterogêneas em vetores de mesma dimensionalidade, torna-se possível treinar modelos sobre uma representação unificada, capaz de generalizar predições para arquiteturas distintas e mitigar as discrepâncias dimensionais e estatísticas entre conjuntos de dados heterogêneos.

Uma barreira adicional ao uso de *encoding* é que, ao projetar representações de motores com geometrias muito distintas para um espaço latente de dimensão fixa, pode haver perda ou distorção de informações relevantes. Essa redução imperfeita de variáveis complexas compromete a fidelidade dos vetores gerados, o que pode prejudicar a capacidade do modelo de capturar características específicas de cada arquitetura de IPMSM.

1.1 Objetivos

O principal objetivo deste projeto é avaliar o potencial do Aprendizado Federado na construção de modelos preditivos avançados, capazes de generalizar para diferentes geometrias de motores elétricos, ao mesmo tempo em que preservam a confidencialidade das informações sensíveis geradas por múltiplos dispositivos e fontes. A proposta adota uma arquitetura de treinamento descentralizada, na qual os dados permanecem localmente em cada nó, sendo compartilhados apenas os parâmetros atualizados do modelo.

A eficácia dos modelos federados será mensurada com base em três métricas de erro: MAE (*Mean Absolute Error*), MSE (*Mean Squared Error*) e MAPE (*Mean Absolute Percentage Error*). Os resultados obtidos serão comparados com aqueles de uma abordagem centralizada de referência (Shimizu, 2022), permitindo avaliar se o Aprendizado Federado mantém ou supera a qualidade preditiva dos métodos tradicionais, além de confirmar suas vantagens em termos de privacidade e escalabilidade.

1.1.1 Objetivos específicos

1. Desenvolver modelos por abordagens centralizadas com redes neurais, avaliando a eficácia da técnica no domínio do problema e comparando o desempenho com os resultados de referência do modelo *XGBoost*.

2. Investigar técnicas de *transfer learning* para verificar o grau de correlação e transferibilidade entre os domínios dos diferentes *datasets*.

3. Mapear as três bases de dados(2D, V e Nabla) para um espaço latente de dimensão comum, de modo a treinar modelos com maior capacidade de generalização entre os conjuntos de dados.

4. Desenvolver e testar diferentes modelos de aprendizado de máquina dentro da arquitetura de aprendizado federado, comparando-os quanto à precisão. Este objetivo tem como meta atingir um desempenho semelhante ou superior ao de modelos centralizados implementados anteriormente (SHIMIZU, 2023).

5. Investigar os principais desafios, limitações e barreiras técnicas associadas à implementação do aprendizado federado na modelagem de desempenho de motores elétricos. Isso inclui aspectos como estratégias de particionamento dos dados em ambientes simulados.

6. Avaliar de que forma a coleta e o uso de dados provenientes de diferentes fontes como variados tipos de motores elétricos e condições operacionais distintas influenciam a eficácia dos modelos treinados por meio do aprendizado federado.

2 Revisão Bibliográfica

Este capítulo tem como objetivo apresentar os fundamentos teóricos e os referenciais utilizados no desenvolvimento deste trabalho. Na Seção 2.1, são abordados os principais conceitos e técnicas relevantes para a compreensão do tema proposto, servindo como base para o entendimento das etapas seguintes do projeto. Já na Seção 2.2, são discutidos os trabalhos correlatos, com o intuito de contextualizar a pesquisa no estado da arte e evidenciar contribuições e lacunas exploradas neste estudo.

2.1 Fundamentação Teórica

2.1.1 Redes Neurais

As redes neurais artificiais surgiram a partir da tentativa de simular, de forma computacional, o funcionamento do cérebro humano. Essa ideia foi inicialmente proposta por *McCulloch* e *Pitts* (MCCULLOCH; PITTS, 1943), em 1943, com a criação de um modelo lógico de neurônio artificial (ABRAHAM, 2002). A proposta consistia em representar matematicamente a atividade neural por meio de unidades simples interconectadas.

O primeiro modelo funcional com capacidade de aprendizado foi o Perceptron 2.2, desenvolvido por *Frank Rosenblatt* em 1958. Essa rede conseguia ajustar seus pesos com base em exemplos, inaugurando o conceito de aprendizado supervisionado ((YADAV; YADAV; KUMAR, 2015)). Embora inicialmente limitada, essa abordagem evoluiu significativamente ao longo das décadas seguintes.

Hoje, as redes neurais são amplamente aplicadas em diversos campos da inteligência artificial, como reconhecimento de imagens, tradução automática, processamento de linguagem natural, sistemas autônomos e diagnósticos médicos. Os avanços recentes em aprendizado profundo possibilitaram o desenvolvimento de modelos extremamente precisos e eficientes, superando, em muitas tarefas, o desempenho humano (WANG; RAJ, 2017).

2.1.2 Neurônios e camadas

O neurônio artificial é o componente fundamental dessas redes. Ele simula, de maneira simplificada, a estrutura de um neurônio biológico ao receber sinais de entrada, processá-los e gerar uma saída. Cada entrada recebida é associada a um peso, que indica sua importância relativa no processamento, e os valores combinados são posteriormente utilizados em um mecanismo de decisão (GOODFELLOW; BENGIO; COURVILLE, 2016).

Inicialmente, os neurônios artificiais foram idealizados como elementos simples, capa-

zes de tomar decisões binárias com base em um limiar. Com o tempo, esses modelos evoluíram para incorporar mecanismos mais complexos, como as funções de ativação, que permitem às redes lidar com relações não lineares e aprender representações mais abstratas dos dados (GOODFELLOW; BENGIO; COURVILLE, 2016).

Figura 2.1 – Neurônio.

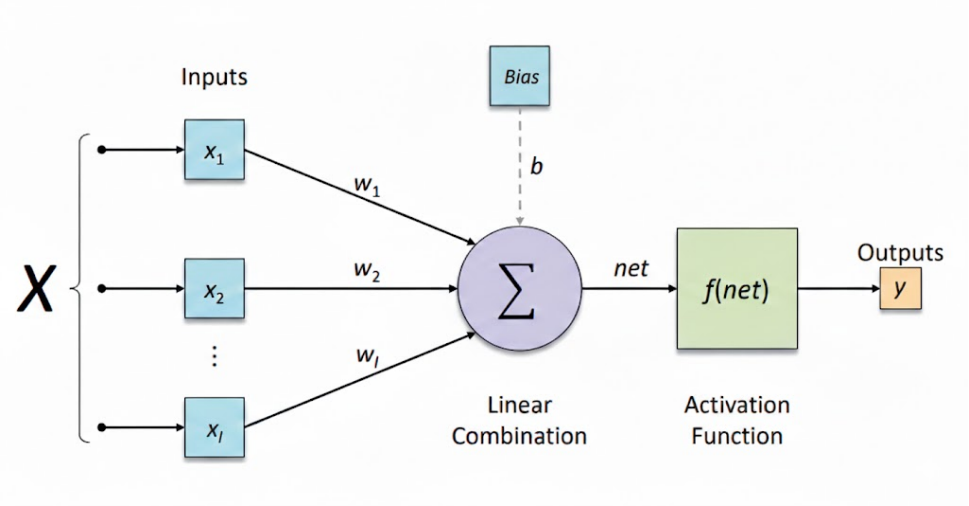


Figura 2.2 – Fonte: (PARMEZAN et al., 2019).

As redes neurais 2.4 são compostas por conjuntos de neurônios organizados em camadas. A primeira delas, conhecida como camada de entrada, é responsável por receber os dados iniciais e repassá-los ao restante da rede. Em seguida, estão as camadas ocultas, que realizam o processamento intermediário. Nessas camadas, os dados são transformados e refinados por meio de múltiplas combinações lineares seguidas da aplicação de funções não lineares, permitindo à rede aprender padrões, relações e estruturas complexas nos dados. Finalmente, a camada de saída é responsável por fornecer a resposta final do modelo, que pode ser, por exemplo, uma classificação, uma previsão numérica ou uma tarefa de regressão. Esse processo de transformação depende diretamente das funções de ativação, que determinam como cada neurônio ativa sua saída a partir do sinal recebido, tema abordado a seguir.(NIELSEN, 2015).

Figura 2.3 – Exemplo de rede simples.

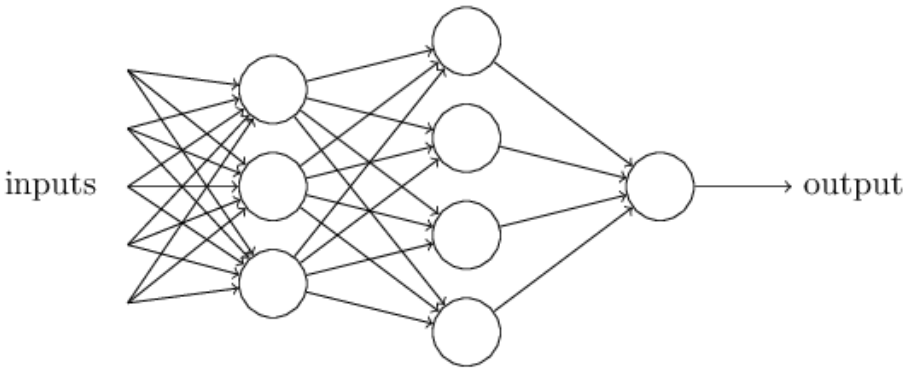


Figura 2.4 – Fonte: (NIELSEN, 2015)

2.1.3 Funções de Ativação

As funções de ativação desempenham um papel fundamental nas redes neurais artificiais ao introduzirem não linearidade no modelo, permitindo que a rede aprenda e represente relações complexas entre as entradas e saídas. Sem essas funções, mesmo redes com múltiplas camadas se comportariam como um modelo linear, limitando sua capacidade de generalização (GOODFELLOW; BENGIO; COURVILLE, 2016).

Uma função de ativação recebe como entrada o resultado da combinação ponderada dos sinais de entrada de um neurônio e determina a saída que será propagada para os próximos neurônios da rede. Entre as funções mais comuns estão a função sigmoide 2.5, que mapeia os valores para o intervalo (0, 1), a tangente hiperbólica 2.6, que retorna valores entre -1 e 1, e a ReLU (Rectified Linear Unit) 2.7, amplamente utilizada por sua simplicidade computacional e eficácia em redes profundas. A escolha da função de ativação impacta diretamente o desempenho da rede, afetando aspectos como convergência, capacidade de modelagem não linear e propagação de gradientes (GOODFELLOW; BENGIO; COURVILLE, 2016).

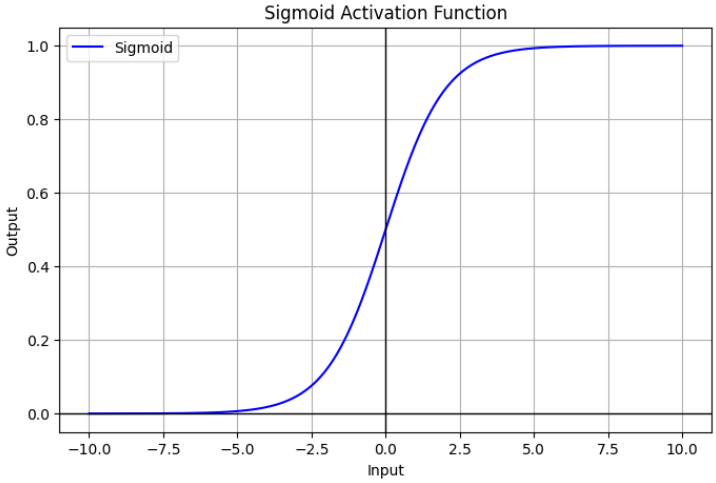


Figura 2.5 – Fonte: (NIELSEN, 2015).

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

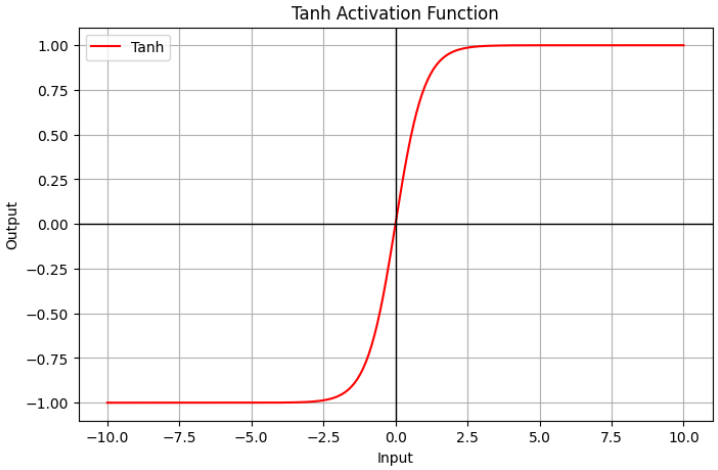


Figura 2.6 – Fonte:. (NIELSEN, 2015)

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.2}$$

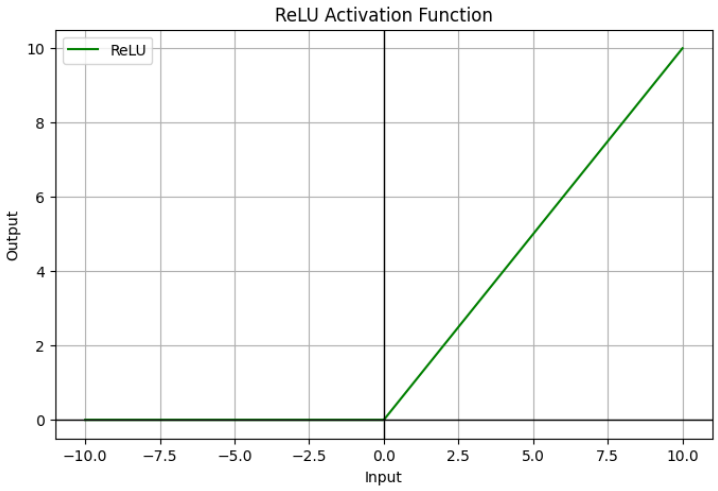


Figura 2.7 – Fonte:. (NIELSEN, 2015)

$$\mathrm{ReLU}(x) = \max(0, x) \tag{2.3}$$

2.1.4 Gradiente

Em redes neurais, o gradiente corresponde às derivadas da função de perda em relação aos parâmetros do modelo (pesos e vieses). Em termos intuitivos, ele quantifica como e quanto

o erro varia diante de pequenas alterações nesses parâmetros, fornecendo uma indicação da direção e da magnitude do ajuste necessário para reduzir a perda. Assim, gradientes com maior magnitude indicam maior sensibilidade da perda às variações do parâmetro, enquanto gradientes próximos de zero sugerem baixo impacto na redução do erro (GOODFELLOW; BENGIO; COURVILLE, 2016).

A partir desse conceito, o treinamento de redes neurais pode ser descrito como um ciclo de cálculo e uso de gradientes. Inicialmente, a rede executa a propagação direta (*forward propagation*), na qual as entradas são transformadas camada a camada até a obtenção da saída e do valor de perda. Em seguida, aplica-se a retropropagação (*backpropagation*), responsável por calcular as derivadas da perda em relação aos parâmetros e propagá-las no sentido inverso da rede, permitindo atualizar pesos e vieses de forma sistemática e otimizar o desempenho do modelo (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.1.5 *Backpropagation* e *Foward propagation*

A *forward propagation*, ou propagação direta, é a primeira fase do processo de treinamento de redes neurais artificiais supervisionadas. Nessa etapa, os dados de entrada percorrem a rede, passando por cada camada e sendo transformados pelas funções de ativação, até gerar uma saída. Essa saída é então comparada com o valor esperado, permitindo o cálculo do erro. A propagação direta é essencial para que o modelo produza uma previsão com base nos pesos e viés atual.

O algoritmo de *backpropagation* é responsável por calcular, de forma sistemática, o gradiente do erro em relação aos pesos e vieses da rede. Esse gradiente é então utilizado por otimizadores, como o ADAM, para atualizar os parâmetros do modelo e reduzir a diferença entre a saída prevista e a esperada. Por meio do gradiente descendente, os ajustes são realizados de maneira eficiente, permitindo o aprendizado em arquiteturas com múltiplas camadas e consolidando-se como um dos pilares do treinamento de redes neurais profundas modernas (BENGIO; COURVILLE; VINCENT, 2013).

2.1.6 *Embedding*

2.1.6.1 Introdução

Representação por *embedding* é uma técnica que permite transformar dados simbólicos ou categóricos como palavras, itens, usuários ou entidades em vetores densos de números reais em um espaço latente de dimensionalidade diferente do original. Essa transformação é fundamental para que algoritmos de aprendizado de máquina possam lidar com dados originalmente não numéricos, convertendo-os para uma forma mais adequada ao processamento computacional (MIKOLOV et al., 2013).

2.1.6.2 *Autoencoders*

Os *autoencoders* são um tipo especial de rede neural artificial projetado para aprender representações compactas e significativas dos dados de entrada. Eles são amplamente utilizados em tarefas como redução de dimensionalidade, transformação de espaço latente e aprendizado não supervisionado. A arquitetura fundamental de um *autoencoder* é composta por duas partes principais: o codificador (*encoder*), responsável por mapear os dados de entrada para uma representação latente de dimensão diferente, e o decodificador (*decoder*), que tenta reconstruir a entrada original a partir dessa representação comprimida. O processo de treinamento consiste em minimizar a diferença entre os dados de entrada e sua reconstrução, geralmente utilizando métricas como o erro quadrático médio. Ao obrigar a rede a reproduzir os dados com menos informação explícita, o *autoencoder* é capaz de extrair características relevantes e eliminar redundâncias, sendo útil em aplicações como detecção de anomalias, compressão, visualização e pré-processamento para outras tarefas de aprendizado (HINTON; SALAKHUTDINOV, 2006).

No contexto deste trabalho, utilizam-se *autoencoders* e técnicas de *embedding* para projetar diferentes bases de dados heterogêneas em um espaço latente comum. Essa abordagem visa reduzir as discrepâncias estruturais entre os conjuntos de dados, permitindo que os modelos operem de forma unificada e comparável, ao mesmo tempo em que preservam ao máximo as características informativas originais de cada domínio.

2.1.7 Transfer Learning

Transfer learning (PAN; YANG, 2010), ou aprendizado por transferência, é uma abordagem de aprendizado de máquina na qual o conhecimento adquirido ao resolver uma tarefa é reutilizado em uma tarefa diferente, mas relacionada. Essa técnica é especialmente útil em cenários onde se dispõe de uma grande quantidade de dados rotulados para uma tarefa-fonte, mas apenas um número limitado de dados para a tarefa-alvo. Em redes neurais profundas, o *transfer learning* é comumente aplicado reutilizando camadas já treinadas geralmente as iniciais, responsáveis por extrair características gerais dos dados enquanto apenas as camadas finais são ajustadas para o novo problema. Essa estratégia permite reduzir o tempo de treinamento, evitar *overfitting* e melhorar o desempenho, principalmente em domínios com dados escassos. Modelos pré-treinados tornaram-se recursos amplamente utilizados na prática do *transfer learning*, sendo aplicados em áreas como visão computacional e processamento de linguagem natural (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.1.8 *Federated Learning*

2.1.8.1 Conceito e Motivação

O *Federated Learning* (FL) (MCMAHAN et al., 2016), ou aprendizado federado, é um paradigma de aprendizado de máquina distribuído que permite treinar modelos colaborativa-

mente sem a necessidade de centralizar os dados. Ao invés de transferir os dados para um servidor central, o FL propõe que o treinamento ocorra localmente nos dispositivos ou instituições que detêm os dados, como *smartphones*, sensores, hospitais ou bancos. Apenas os parâmetros do modelo treinado localmente como pesos ou gradientes são enviados ao servidor central para serem agregados, preservando a privacidade dos dados originais.

A motivação central por trás do FL está na crescente preocupação com a privacidade de dados, limitações de largura de banda e a dificuldade de centralizar grandes volumes de dados heterogêneos. Esse modelo é particularmente vantajoso em ambientes onde os dados são sensíveis ou descentralizados por natureza, como em sistemas de saúde, redes móveis ou ambientes financeiros. Segundo McMahan et al. (MCMAHAN et al., 2016), o FL surgiu como uma alternativa eficiente para treinar redes neurais profundas, respeitando as restrições de privacidade e infraestrutura.

2.1.8.2 Arquitetura e Funcionamento

A arquitetura do *Federated Learning* é composta, tipicamente, por um servidor central e um conjunto de clientes (dispositivos ou organizações) que colaboram no treinamento de um modelo global. O processo inicia-se com o envio de um modelo inicial do servidor para um subconjunto de clientes. Cada cliente treina o modelo localmente com seus próprios dados privados e envia ao servidor apenas as atualizações dos parâmetros, como gradientes ou pesos.

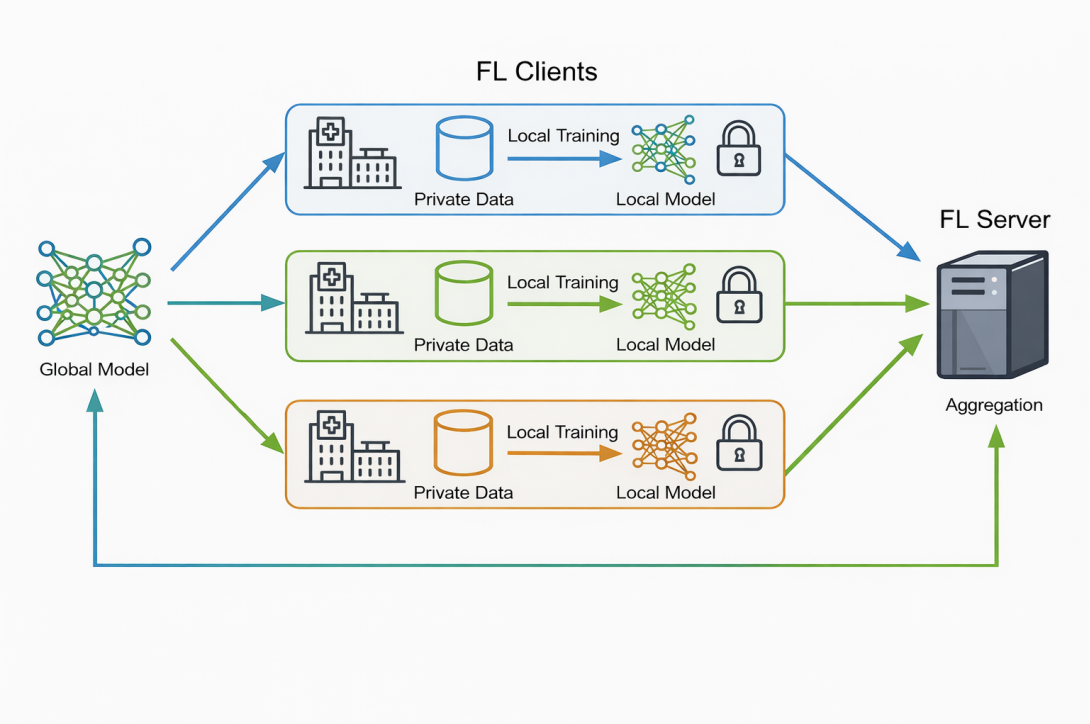


Figura 2.8 – Diagrama do Aprendizado Federado adaptado conceitualmente de (NVIDIA, 2024)

2.1.8.3 *Federated Averaging*

O algoritmo *Federated Averaging* (FedAvg) 2.8, proposto por McMahan et al. (MCMAHAN et al., 2016), é o método de agregação mais amplamente utilizado nesse contexto. Ao final de cada rodada de treinamento local, os clientes enviam ao servidor os pesos atualizados w_k^t e a quantidade de amostras n_k utilizada. O servidor então atualiza o modelo global w^{t+1} por meio de uma média ponderada:

$$w^{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \cdot w_k^t \tag{2.4}$$

em que K é o número total de clientes, n_k representa o número de amostras do cliente k e $n = \sum_{k=1}^{K} n_k$ é o total de dados envolvidos. Esse ciclo é repetido até que o modelo alcance um desempenho satisfatório.

Durante todo o processo, os dados permanecem nos dispositivos dos clientes, o que favorece a privacidade e reduz riscos de exposição. Contudo, a heterogeneidade estatística dos dados, bem como variações em hardware e conectividade, podem comprometer a eficiência e a convergência do modelo global (ZHANG et al., 2021).

2.1.9 Motores Síncronos com Ímãs Permanentes Internos (IPMSM)

Os *Interior Permanent Magnet Synchronous Motors* (IPMSMs) (BOLDEA et al., 2014) são motores síncronos com ímãs permanentes embutidos na estrutura do rotor. Essa configuração estrutural difere dos motores de ímãs montados superficialmente (SPMSMs) e oferece vantagens significativas, especialmente para aplicações em veículos elétricos e híbridos. Ao integrar os ímãs no interior do rotor, os IPMSMs permitem a exploração do torque de relutância gerado pela saliência magnética além do torque fornecido pelos próprios ímãs, resultando em maior densidade de potência e eficiência energética.

No presente trabalho, o modelo de IPMSM é adotado devido ao seu alto potencial de desempenho em termos de eficiência e controle dinâmico. Contudo, o processo de projeto e análise desses motores envolve simulações complexas, como análise por elementos finitos (FEA), para diferentes condições de operação. Para mitigar esse custo computacional, o uso de técnicas de aprendizado de máquina, como modelos baseados em *XGBoost*, é explorado com o objetivo de prever o comportamento magnético e térmico do motor de forma precisa, reduzindo significativamente o tempo de otimização do sistema (SHIN et al., 2014).

2.2 Trabalhos Relacionados

Shimizu (2023) propôs um modelo de otimização da eficiência de motores síncronos de ímãs permanentes internos (IPMSM), com foco na aplicação de aprendizado de máquina para

aprimorar o controle e a operação desses motores em aplicações automotivas. O estudo desenvolve um modelo baseado em modelos substitutos, utilizando algoritmos de aprendizado, como *XGBoost*, para prever de maneira eficiente o comportamento magnético e térmico dos motores sem a necessidade de longas simulações por elementos finitos (FEA). Essa abordagem reduz significativamente o tempo de análise e permite a otimização de parâmetros, como a geometria do rotor e as condições operacionais, para maximizar a eficiência e minimizar perdas.

O autor explora três topologias de rotor distintas (2D, Nabla e V) para avaliar como diferentes arranjos geométricos podem impactar a eficiência do motor em diferentes condições de operação. A abordagem proposta não só melhora a precisão da previsão das perdas por histerese e correntes parasitas, mas também assegura que o processo de otimização seja mais acessível, permitindo a personalização do motor para aplicações específicas. O trabalho se destaca pelo uso de técnicas de aprendizado de máquina para aprimorar o design e controle de motores elétricos, demonstrando seu grande potencial para aplicações em veículos eletrificados, onde a eficiência energética é crucial.

O artigo "Investigation of Dynamic Eccentricity in Interior Permanent Magnet Synchronous Motor through Finite Element Method and Machine Learning"(SHEN; MA; LI, 2025)g investiga o impacto da excentricidade dinâmica em um Motor Síncrono de Imã Permanente Interno (IPMSM) utilizando a análise de elementos finitos (FEA) e técnicas de aprendizado de máquina. O foco é na detecção de falhas causadas pela excentricidade dinâmica, um problema comum em motores elétricos que pode prejudicar seu desempenho.

No estudo, foi utilizado um modelo de 550 W e 220 V do IPMSM no software ANSYS Maxwell, com falhas de excentricidade variando de 10% a 40%. As características do motor, como a corrente do estator e a densidade de fluxo radial, foram comparadas entre condições normais e excêntricas. Os dados sobre a corrente do enrolamento e o fluxo de ar (componente radial) foram obtidos e analisados utilizando ferramentas de aprendizado de máquina no MATLAB. A técnica de ensemble bagged trees foi empregada para detectar falhas, apresentando uma precisão de 82,5% para a corrente do estator.

Além disso, a excentricidade dinâmica foi identificada com uma precisão de 98,5% usando os dados de densidade de fluxo radial, demonstrando que abordagens de aprendizado de máquina podem ser eficazes na detecção de falhas em IPMSMs. O estudo destaca a utilidade do aprendizado de máquina para identificar falhas com base nas variáveis do motor, como a corrente do estator e o fluxo radial.

O trabalho também discute a importância da detecção precoce de falhas para melhorar a confiabilidade e o desempenho dos motores em aplicações industriais e de tração.

O trabalho de *YOU* (YOU, 2020) investiga o uso de deep learning como metamodelo para o projeto ótimo de motores síncronos de ímãs permanentes (PMSM) em aplicações automotivas. O problema é formulado como uma otimização multiobjetivo da geometria do rotor: a partir de

alguns parâmetros de projeto (como o ângulo entre ímãs em V e a espessura da nervura), o autor treina modelos preditivos (MLP e *Kriging*) para aproximar resultados de simulações por elementos finitos e, em seguida, acopla esses modelos a um algoritmo meta-heurístico híbrido para buscar configurações que maximizem o torque médio e minimizem a distorção harmônica da força contraeletromotriz, sob restrições de eficiência e ondulação de torque. Assim, o foco está em acelerar o ciclo de projeto geométrico de um PMSM específico, substituindo parte das simulações FEA por um modelo aproximador centralizado.

Em contraste, este trabalho não aborda diretamente a etapa de síntese geométrica do motor, mas parte de um conjunto já existente de designs de IPMSM e utiliza machine learning para modelar o comportamento eletromagnético (como perdas em diferentes componentes do motor) a partir de bases de dados heterogêneas. Além disso, enquanto *You* (YOU, 2020) considera um cenário estritamente centralizado, aqui investiga-se o uso de Aprendizado Federado para treinar modelos a partir de dados distribuídos entre diferentes clientes (por exemplo, diferentes topologias de motor ou conjuntos de simulação), preservando a separação dos dados e estudando os impactos da distribuição no desempenho preditivo. Dessa forma, o trabalho de *You* (YOU, 2020) é complementar: mostra o potencial de metamodelos baseados em redes neurais para apoiar o projeto de PMSM, enquanto esta monografia explora, em nível teórico e experimental, como técnicas de Aprendizado Federado podem ser incorporadas à modelagem de motores elétricos em cenários distribuídos.

Al-Gabalawy et al. (AL-GABALAWY et al., 2022) investigam a predição de temperatura em motores síncronos de ímã permanente (PMSM) aplicados a veículos elétricos, com foco específico na estimativa da temperatura do rotor a partir de medições acessíveis em operação, como temperaturas ambiente e do fluido de arrefecimento, tensões u_d e u_q, correntes i_d e i_q, velocidade e torque do motor. Os autores constroem uma base experimental extensa, realizam uma análise exploratória detalhada (distribuições, correlações, *outliers*) e comparam diferentes modelos supervisionados, incluindo regressão linear e polinomial com regularização (Ridge e Lasso), *Support Vector Regression* (SVR) e métodos baseados em árvores e *boosting* (*XGBoost* e variantes). A partir dessa comparação, o estudo discute, em termos de ajuste estatístico e comportamento dos resíduos, quais algoritmos capturam melhor as relações não lineares entre as variáveis elétricas/mecânicas e a temperatura, identificando ainda quais atributos (principalmente ambiente, *coolant* e velocidade) exercem maior influência sobre a temperatura do ímã permanente.

Em relação à presente monografia, Al-Gabalawy et al. (AL-GABALAWY et al., 2022) posiciona-se como uma abordagem de monitoramento térmico voltada para a operação segura do PMSM em EVs, tratando a predição de temperatura como um problema de regressão centralizado em um único *dataset* experimental. Aqui, por outro lado, o foco não está na modelagem térmica em si, mas na utilização de técnicas de *machine learning* (e, em particular, de Aprendizado Federado) para modelar grandezas eletromagnéticas de motores (como perdas, flu-

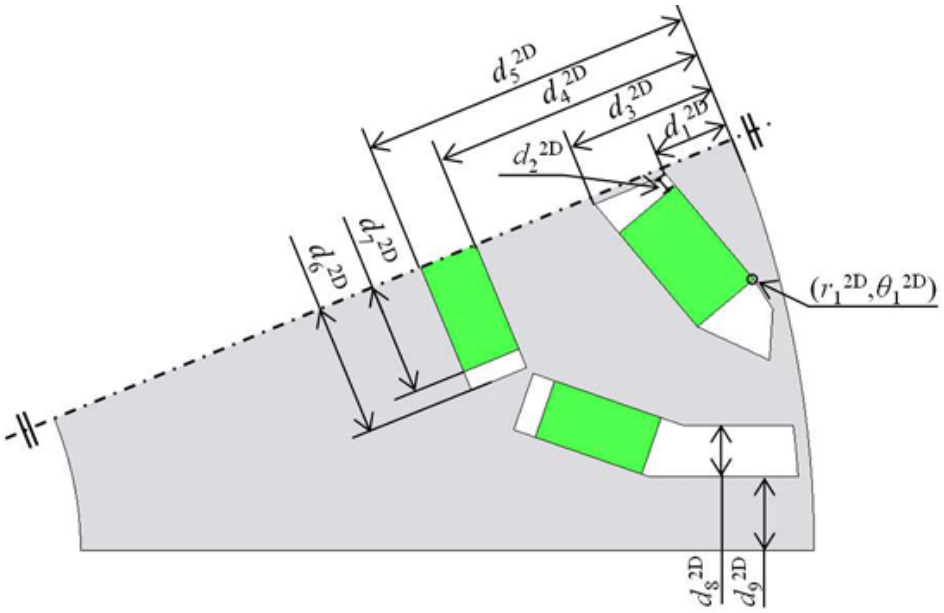
xos ou parâmetros equivalentes) a partir de bases de dados distribuídas entre diferentes clientes (por exemplo, distintas topologias ou cenários de operação). Enquanto *Al-Gabalawy et al.* (AL-GABALAWY et al., 2022) exploram principalmente modelos clássicos de regressão e ensemble para um PMSM específico, este trabalho discute, em níveis teóricos e experimentais, como arquiteturas de aprendizado distribuído podem ser empregadas para treinar modelos em múltiplos conjuntos de dados, preservando a separação dos dados e analisando o impacto dessa distribuição no desempenho preditivo o que torna os dois estudos complementares no uso de aprendizado de máquina aplicado a motores de ímã permanente em veículos elétricos.

3 Desenvolvimento

3.1 Preparando os dados

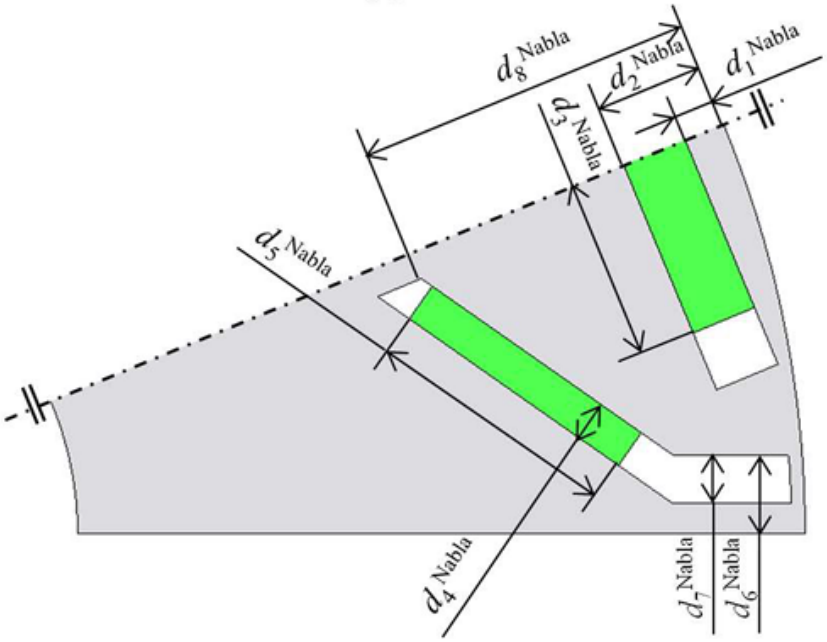
O projeto utilizou a base de dados correspondente às três topologias de rotor descritas no artigo de Yuki Shimizu (SHIMIZU, 2023): 2D (3.1), Nabla (3.2) e V (3.3). Cada uma dessas bases contém registros gerados a partir de simulações por Análise de Elementos Finitos (FEA), que foram empregadas para modelar e avaliar o desempenho de motores síncronos de ímã permanente de rotor interno (IPMSM). Os parâmetros (*inputs*) são: parâmetros geométricos do motor, condições de operação representadas pela velocidade de rotação, componentes da corrente elétrica Id (eixo direto) e Iq (eixo em quadratura) e coordenadas polares (r1, t1). Os alvos (*targets*) são: torque calculado e perdas por histerese.

Figura 3.1 – Topologia do Motor 2D (SHIMIZU, 2023).



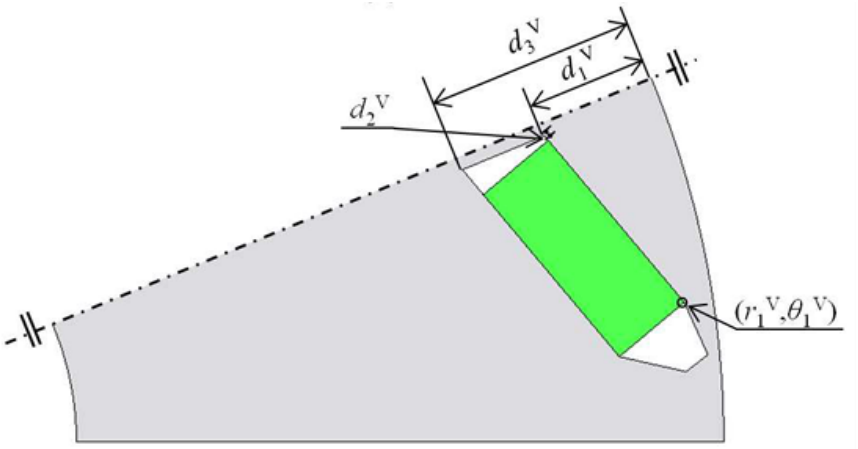
Fonte: (SHIMIZU, 2023).

Figura 3.2 – Topologia do Motor Nabla (SHIMIZU, 2023).



Fonte: (SHIMIZU, 2023).

Figura 3.3 – Topologia do Motor V (SHIMIZU, 2023).



Fonte: (SHIMIZU, 2023).

Originalmente, as informações de cada topologia estavam dispersas em múltiplos arquivos independentes, provenientes das execuções de FEA utilizadas no artigo de referência (SHIMIZU, 2023), o que demandou uma etapa preliminar de consolidação. Dessa forma, esses arquivos foram agregados, validados e reorganizados em um único conjunto de dados por topologia, assegurando a integridade, a rastreabilidade e a consistência dos registros. Na sequência, realizou-se a normalização de todas as variáveis, com o propósito de padronizar as escalas numéricas, mitigar o efeito de discrepâncias de magnitude entre atributos e favorecer a estabilidade e a convergência dos algoritmos de aprendizado de máquina empregados nas etapas posteriores.

3.2 Métodos de avaliação

Nos experimentos realizados, serão adotadas três métricas de avaliação de erro: *Mean Squared* Error (MSE), *Mean Absolute Percentage Error* (MAPE) e *Mean Absolute Error* (MAE). Essas métricas foram escolhidas principalmente por conta do artigo de referência (SHIMIZU, 2023) que também usa essas métricas para avaliação.

3.2.1 *Mean Squared Error* (MSE)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.1}$$

onde n é o número de observações, y_i é o valor real e $\hat{y}_i$ é o valor previsto pelo modelo (BOTCHKAREV, 2019).

O MSE (Erro Quadrático Médio) mede a média dos quadrados das diferenças entre os valores previstos pelo modelo e os valores reais. Por elevar ao quadrado os erros antes de calcular a média, o MSE penaliza de forma mais intensa os erros grandes, tornando-o particularmente sensível a discrepâncias acentuadas. Essa característica o torna útil quando se deseja evitar previsões com grandes desvios individuais, mas também faz com que seja mais influenciado por *outliers* (BOTCHKAREV, 2019).

3.2.2 *Mean Absolute Percentage Error* (MAPE)

$$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{3.2}$$

onde n é o número de observações, y_i é o valor real e $\hat{y}_i$ é o valor previsto pelo modelo. O resultado é expresso em porcentagem (BOTCHKAREV, 2019).

O MAPE (Erro Percentual Absoluto Médio) calcula a média dos erros absolutos em termos percentuais, dividindo a diferença entre previsão e valor real pelo próprio valor real e multiplicando por 100. Essa métrica fornece uma interpretação intuitiva, expressa como porcentagem de erro médio em relação aos valores observados. No entanto, o MAPE pode se tornar problemático quando os valores reais se aproximam de zero, pois o denominador pequeno amplifica o erro percentual (BOTCHKAREV, 2019).

3.2.3 *Mean Absolute Error* (MAE)

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.3}$$

onde n é o número de observações, y_i é o valor real e $\hat{y}_i$ é o valor previsto pelo modelo (BOTCHKAREV, 2019).

O MAE (Erro Absoluto Médio) representa a média das diferenças absolutas entre as previsões e os valores reais. Diferente do MSE, ele não eleva os erros ao quadrado, o que faz com que todos os desvios tenham peso proporcional, sem penalizar excessivamente os maiores erros. É uma métrica de interpretação simples e direta, expressa nas mesmas unidades da variável-alvo, e é menos sensível a *outliers* em comparação com o MSE (BOTCHKAREV, 2019).

A avaliação dos modelos foi realizada com divisão de treino, validação e teste, sendo uma proporção de 70% treino, 10% validação e 20% teste.

3.3 Metodologia

No projeto, é adotada uma sequência estruturada de experimentos com diferentes arquiteturas de redes neurais, a fim de validar hipóteses e explorar características específicas da base de dados. Inicialmente, os conjuntos de dados de cada topologia são submetidos a modelos simples de redes neurais densas (*fully connected*), contendo poucas camadas e número reduzido de neurônios. Essa etapa tem como objetivo principal avaliar a complexidade do problema, identificar padrões iniciais e obter uma referência de desempenho para comparações futuras.

Em seguida, é realizada uma série de experimentos com redes neurais densas de diferentes tamanhos e profundidades, buscando arquiteturas mais refinadas e ajustadas para cada tarefa específica.

O próximo estágio envolve a aplicação de *transfer learning* entre topologias distintas. Modelos previamente treinados em uma das bases de dados são reutilizados e ajustados para outra, com o objetivo de investigar a existência de correspondências semânticas e estruturais entre as diferentes topologias de motores (2D, Nabla e V).

Após essa etapa, entram em cena *autoencoders*, que serão utilizados para projetar as três bases de dados em um espaço latente comum. A partir dessa representação unificada, será possível treinar um único modelo capaz de generalizar de forma eficaz para todas as topologias, preservando a diversidade de características.

Posteriormente, o foco se desloca para o aprendizado federado. Primeiramente, serão treinados modelos em um ambiente federado para avaliar se a federação impacta de forma significativa o desempenho em comparação ao treinamento centralizado.

Por fim, será confeccionado um modelo geral federado, no qual as bases de dados permanecem fisicamente separadas, mas colaboram indiretamente no treinamento, permitindo avaliar a viabilidade de um sistema único que respeite restrições de privacidade e fragmentação de dados, sem perda significativa de desempenho.

3.4 Tecnologias utilizadas

Todo o código do projeto foi desenvolvido em *Python*, utilizando um conjunto de bibliotecas e *frameworks* específicos para cada etapa do fluxo de trabalho. O *PyTorch* foi empregado para a implementação e treinamento das redes neurais, devido à sua flexibilidade e ampla adoção na comunidade de *deep learning*. A manipulação e organização dos dados foram realizadas com pandas, que oferece recursos eficientes para tratamento de tabelas.

Para a divisão dos conjuntos de dados em treino, teste e validação, utilizou-se a função *train_test_split* da biblioteca *scikit-learn* (*sklearn*), garantindo uma separação consistente e reprodutível. A otimização automática de hiperparâmetros foi realizada com *Optuna*. O *Transfer Learning* foi implementado por meio de funções personalizadas para treinamento e congelamento de camadas no *PyTorch*. Por fim, o *framework Flower* foi empregado para implementar o aprendizado federado, permitindo a simulação e avaliação de cenários distribuídos, preservando a separação física das bases de dados.

4 Experimentos

4.1 Experimento 1 - Avaliação completa do do modelo base de (SHIMIZU, 2023)

Esta seção tem como objetivo estimar o MSE, MAE e $MAPE$ para o *XGBoost* proposto por (SHIMIZU, 2023) 4.1, bem como para modelos de redes neurais simples.

Motor	Target	Test MSE	Test MAE	Test MAPE (%)
2D	Hysteresis	0.0183	0.0277	14.15
	Joule	0.1901	0.0333	19.28
	Total	0.3245	0.0589	14.80
Nabla	Hysteresis	0.0045	0.0414	113.59
	Joule	0.0094	0.0525	29.79
	Total	0.0232	0.0867	30.72
V	Hysteresis	0.0024	0.0327	31.38
	Joule	0.0034	0.0352	29.20
	Total	0.0095	0.0621	13.85

Tabela 4.1 – Resultados do Modelo *XGBoost* para Diferentes Motores

Os três motores analisados (2D, Nabla e V) foram testados sob três abordagens distintas. Em cada teste, o modelo foi treinado com o conjunto de dados do respectivo motor, tendo um dos três alvos possíveis: “hysteresis”, “joule” ou “total”. O alvo “total” representa uma junção dos dois primeiros, sendo um alvo no qual os modelos devem prever simultaneamente os dois alvos (“hysteresis” e “joule”).

4.2 Modelos densos

Os primeiros modelos desenvolvidos consistiram em pequenas redes totalmente conectadas, com a quantidade de neurônios variando entre 16 e 64.

Target	Test MSE	Test MAE	Test MAPE (%)
2D Hysteresis	0.0180	0.0227	11.34
2D Joule	0.1897	0.0289	12.45
2D Total	0.1039	0.0262	13.02

Tabela 4.2 – Resultado para o motor 2D

Perdas, Ganhos	Test MSE	Test MAE	Test MAPE
Hysteresis	0.0003	0.0050	2,81
Joule	0.0004	0.0044	6,83
Total	0.2206	0.0327	1,79

Tabela 4.3 – Valores das perdas para 2D

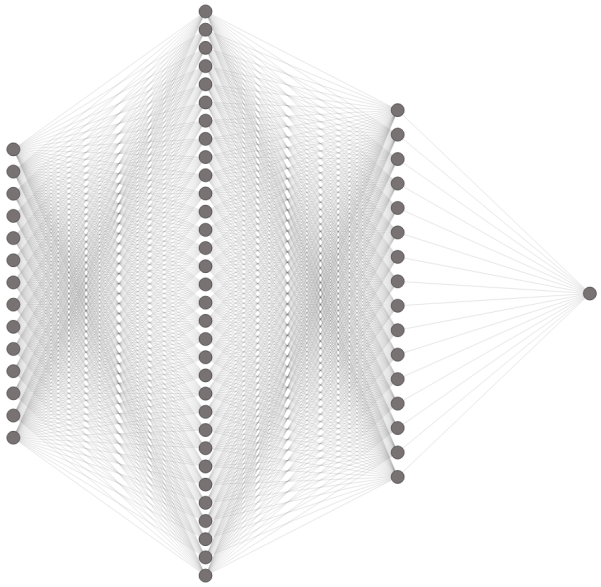


Figura 4.1 – Representação gráfica do modelo 2d

A Tabela 4.2 mostra o resultado alcançado com uma rede neural de três camadas, cada uma contendo, respectivamente, 14, 32 e 16 neurônios e utilizando funções de ativação *ReLU* e MSE como função de perda. O modelo foi treinado por 10.000 épocas e, mesmo com essa estrutura simples, superou o desempenho do *XGBoost* em todas as métricas avaliadas, apresentando uma leve melhora nos resultados.

Figura 4.2 – Curvas de treino e validação

A imagem 4.2 mostra a evolução do treino. Observa-se que, aproximadamente entre as épocas 3000 e 4000, a perda no conjunto de validação oscila em torno do intervalo entre 0,022 e 0,020, indicando que os ganhos de desempenho tornam-se progressivamente menores e o modelo entra em um regime de quase estagnação.

A Tabela 4.13 apresenta a diferença entre os valores obtidos pelo modelo *XGBoost* e a rede neural utilizada no experimento (*XGBoost* - Rede). Nota-se que essa diferença foi positiva, indicando que a rede neural alcançou um desempenho superior ao modelo alvo, especialmente quando ambos os alvos foram combinados no parâmetro "Total".

Target	**Test MSE**	**Test MAE**	**Test MAPE (%)**
Nabla *Hysteresis*	0.0264	0.0921	61.28
Nabla Joule	0.0454	0.1040	101.11
Nabla Total	0.0341	0.0977	80.59

Tabela 4.4 – Resultados para o motor nabla

Perdas, Ganhos	**Test MSE**	**Test MAE**	**Test MAPE**
Hysteresis	0.0219	0.0507	52.31
Joule	0.0360	0.0515	71.32
Total	0.0109	0.0110	49.87

Tabela 4.5 – Valores das perdas para nabla

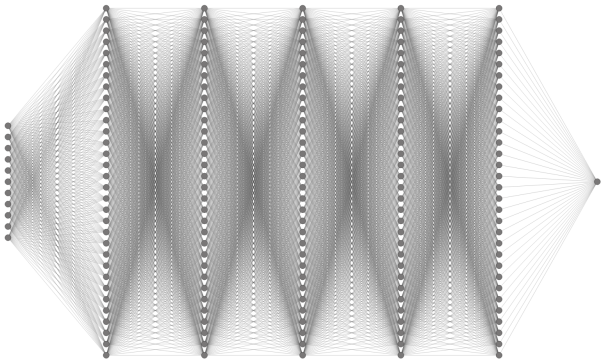


Figura 4.3 – Representação gráfica do modelo Nabla

O modelo foi treinado utilizando uma rede neural de seis camadas, onde a camada de entrada possui 11 neurônios e as camadas subsequentes, 32 neurônios cada, todas empregando a função de ativação ReLU e MSE como função de perda. O treinamento foi realizado ao longo de 50.000 épocas.

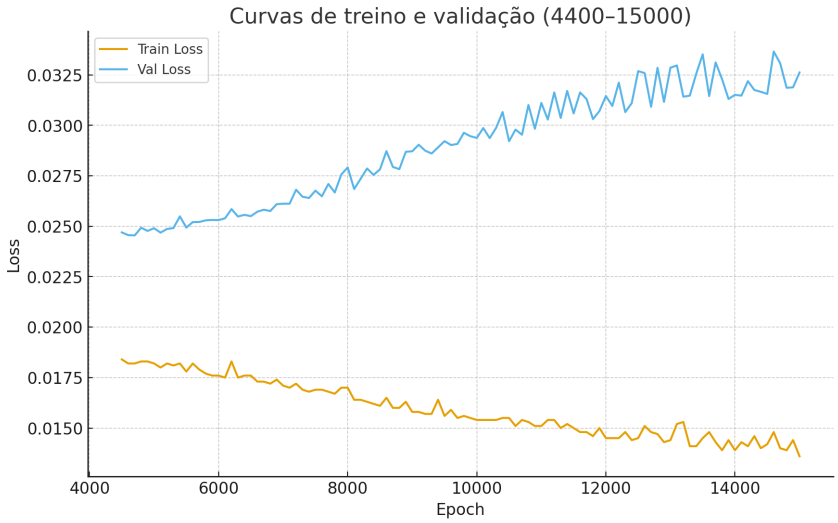


Figura 4.4 – Curvas de treino e validação

A Imagem 4.4 resume o experimento com treino estendido até 50,000 épocas. Nota-se que, embora a perda de treino continue diminuindo, a perda de validação passa a oscilar em torno de 0,0300,040, sem ganhos relevantes, caracterizando sobreajuste e indicando que épocas adicionais não trazem melhorias efetivas ao modelo.

Os resultados apresentados na Tabela 5 evidenciam que os valores exibidos na Tabela 4 apresentam uma diferença negativa considerável em comparação com o modelo *XGBoost*. Isso demonstra que não é possível alcançar métricas satisfatórias utilizando uma rede neural simples, sendo necessária uma investigação mais aprofundada na arquitetura das redes empregadas nesta base de dados.

Target	**Test MSE**	**Test MAE**	**Test MAPE (%)**
V Hysteresis	0.0184	0.0815	54.66
V Joule	0.0289	0.0857	78.56
V Total	0.0237	0.0845	76.38

Tabela 4.6 – Resultados para o motor V

Perdas, Ganhos	**Test MSE**	**Test MAE**	**Test MAPE**
Hysteresis	0.0160	0.0488	23.28
Joule	0.0255	0.0505	49.36
Total	0.0142	0.0224	62.53

Tabela 4.7 – Valores das perdas para V

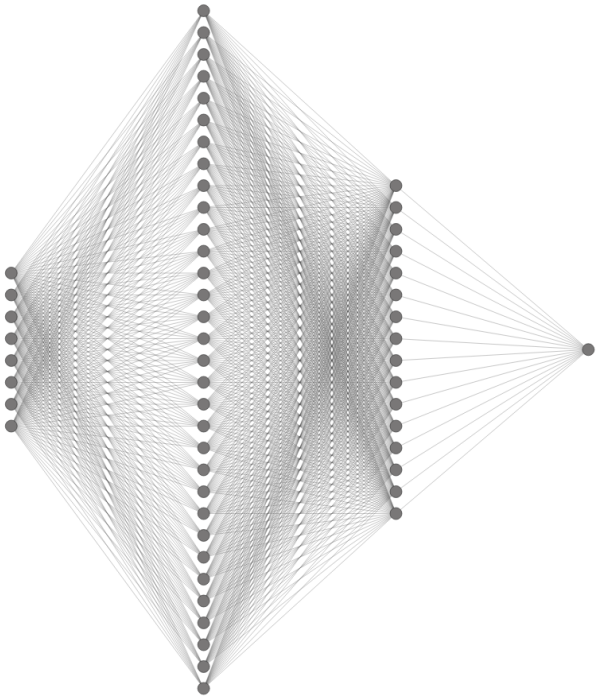


Figura 4.5 – Representação gráfica do modelo V

O modelo foi treinado utilizando uma rede neural de três camadas, cada uma contendo, respectivamente, 8, 32 e 16 neurônios e empregando funções de ativação ReLU, ao longo de 10.000 épocas. Assim como no caso do modelo Nabla, o desempenho dessa rede foi significativamente inferior ao do *XGBoost* em todas as métricas avaliadas.

A Tabela 4.5 revela que todos os valores obtidos pelo modelo foram inferiores aos do *XGBoost*, demonstrando que não foi possível atingir resultados satisfatórios com a configuração atual, que emprega essa quantidade de camadas e neurônios. Isso indica a necessidade de mais experimentos com redes neurais mais extensas e complexas para melhorar o desempenho nesta base de dados.

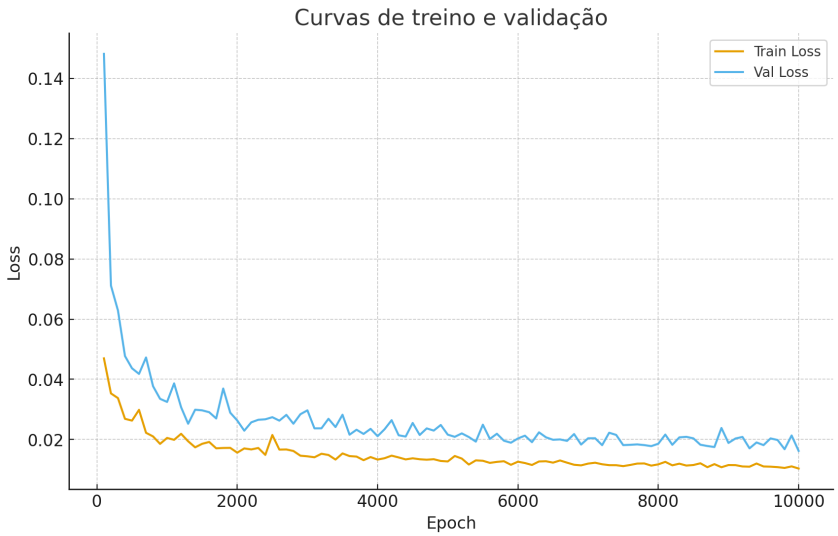


Figura 4.6 – Curvas de treino e validação

A Figura 4.6 mostra que o erro médio absoluto cai rapidamente nas primeiras épocas e, a partir de cerca de 3 000 épocas, entra em regime de estabilização. O *train loss* continua reduzindo lentamente, enquanto o *val loss* oscila em torno de 0,018–0,022, indicando convergência do modelo com boa generalização.

Ao final desta seção, ainda não foi possível concluir se é viável desenvolver modelos específicos para todas as bases de dados que apresentem desempenho comparável ao do *XGBoost*. Testes mais específicos e aprofundados no futuro deverão fornecer uma resposta mais definitiva a essa questão.

4.3 Auto ML

A biblioteca de Auto ML utilizada foi o Optuna. Essa ferramenta permite a execução de múltiplos experimentos, ajustando diferentes configurações de hiperparâmetros. Os modelos testados variaram entre 10 e 40 camadas, com 16 a 64 neurônios por camada, learning rate entre 0.0001 e 0.001, função de ativação ReLU, 1000 epocas de treino e MSE como função de *loss*. Depois dessa etapa, selecionamos apenas os modelos com melhor desempenho e realizamos um novo treino mais intenso, agora com 8000 épocas.

Target	**Test MSE**	**Test MAE**	**Test MAPE** (%)
V Hysteresis	0.000298	0.012242	8.2171
V Joule	0.000488	0.013030	9.4866
V Total	0.000440	0.013041	12.9545

Tabela 4.8 – Melhor modelo obtido para o Motor V

Perdas, Ganhos	**Test MSE**	**Test MAE**	**Test MAPE**
Hysteresis	0.0021	0.0205	23.1
Joule	0.0029	0.0222	19.71
Total	0.0091	0.0491	0.9

Tabela 4.9 – Valores das perdas para V

Neste experimento, foi possível superar os problemas identificados na seção anterior. Conforme demonstrado na Tabela 4.9, a diferença entre os resultados da rede neural e do modelo *XGBoost* tornou-se positiva, evidenciando a superioridade da rede neural. Assim, conclui-se que, com a arquitetura adequada e o número correto de épocas de treinamento, as redes neurais têm potencial para oferecer desempenho superior.

Target	**Test MSE**	**Test MAE**	**Test MAPE** (%)
Nabla Hysteresis	0.001773	0.024602	34.1327
Nabla Joule	0.004840	0.036343	22.1742
Nabla Total	0.003168	0.030273	28.2522

Tabela 4.10 – Resultados dos modelos do motor Nabla

Perdas, Ganhos	Test MSE	Test MAE	Test MAPE
Hysteresis	0.0027	0.0168	79.46
Joule	0.0046	0.0162	7.62
Total	0.0200	0.0564	2.47

Tabela 4.11 – Valores das perdas para Nabla

Como evidenciado na Tabela 4.11, as diferenças foram positivas em comparação com o *XGBoost*, demonstrando que as redes neurais podem alcançar resultados robustos na base de dados Nabla, desde que sejam utilizadas arquiteturas apropriadas e treinadas por um número suficiente de épocas.

Target	Test MSE	Test MAE	Test MAPE (%)
2D Hysteresis	0.017302	0.013864	6.2948
2D Joule	0.188623	0.020588	10.3438
2D Total	0.103242	0.020055	8.6713

Tabela 4.12 – Resultados dos modelos do motor 2D

Perdas, Ganhos	Test MSE	Test MAE	Test MAPE
Hysteresis	0.0010	0.0138	7.8552
Joule	0.0015	0.0127	8.9362
Total	0.2213	0.0388	6.1287

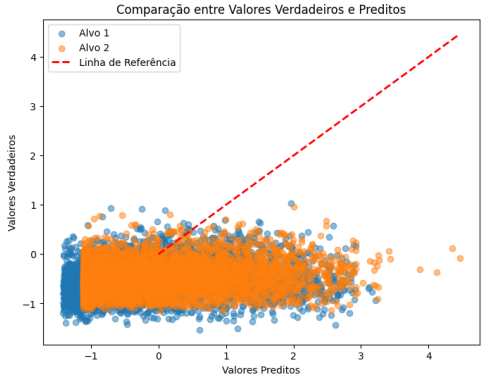
Tabela 4.13 – Valores das perdas para 2D

A Tabela 4.2 evidencia uma melhoria significativa nas predições do modelo 2D em relação aos resultados apresentados na Tabela 4.13, demonstrando que, mesmo em uma base de dados onde já se alcançaram resultados satisfatórios, é possível obter aprimoramentos adicionais ao se utilizar redes maiores e mais complexas.

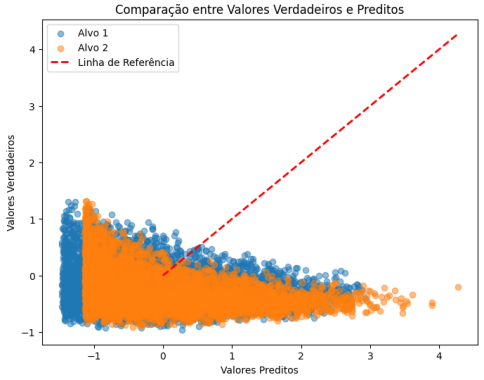
Ao final do experimento, os resultados indicam que as redes neurais representam uma alternativa viável para a realização de predições, especialmente no contexto de modelos centralizados.

4.4 *Transfer Learning*

O primeiro teste realizado teve como objetivo avaliar se um modelo treinado em um determinado conjunto de dados de um motor seria capaz de apresentar um desempenho satisfatório em outro conjunto de dados, sem a aplicação de qualquer tipo de pré-processamento ou *fine-tuning*.



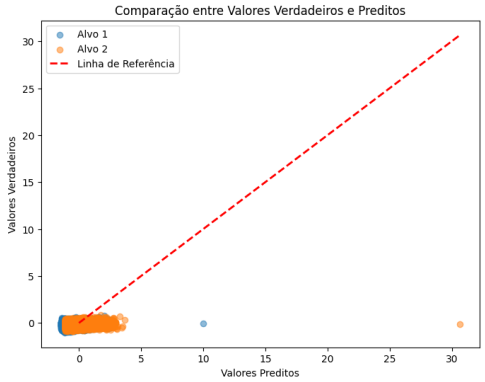
(a) De 2D para Nabla



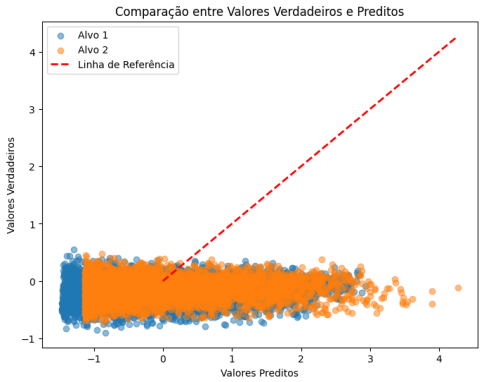
(b) De 2D para V

Figura 4.7 – *Transfer-learning* Nabla. (*Target*: total)

A Figura 4.7 ilustra o desempenho do modelo 2D quando aplicado às bases de dados de Nabla e V. Quanto mais próximos os pontos estiverem da linha de referência, melhores são as predições. No entanto, observa-se que o modelo 2D não está conseguindo generalizar para as outras duas bases de dados.



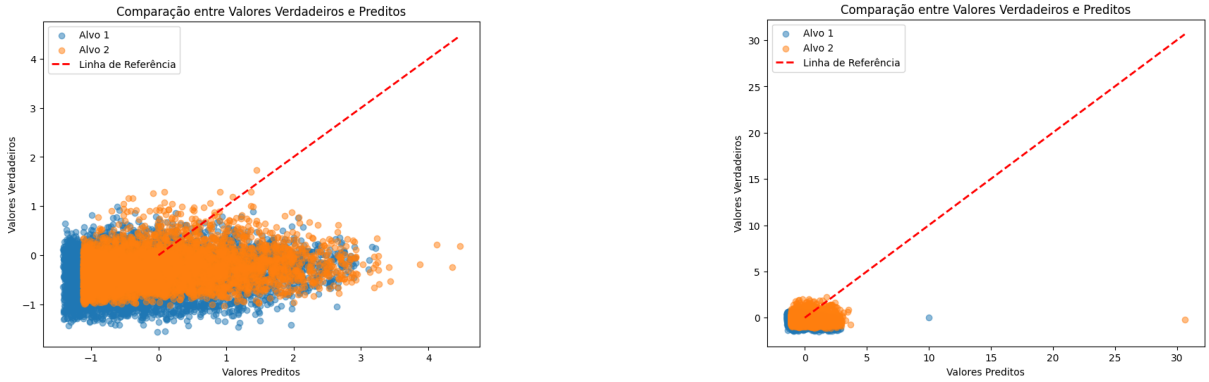
(a) De Nabla para 2D



(b) De Nabla para V

Figura 4.8 – Transfer-learning Nabla. (Target: total)

A Figura 4.8 exibe as predições do modelo Nabla para as bases V e 2D, assim como o modelo 2D, o modelo Nabla não foi capaz de generalizar bem para as outras bases de dados.



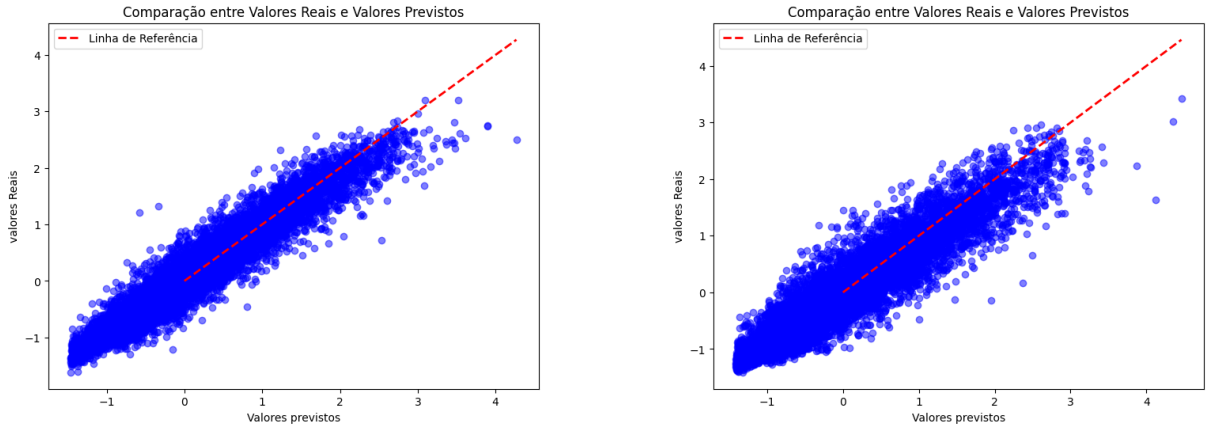
(a) De V para Nabla

(b) De V para 2D

Figura 4.9 – Transfer-learning V. (Target: total)

A Figura 4.9 apresenta as predições do modelo V para as bases de dados Nabla e 2D. Assim como observado nos demais modelos, o modelo V não conseguiu generalizar de forma satisfatória para as outras bases de dados.

Os resultados obtidos indicaram que os modelos não foram capazes de generalizar para outros conjuntos de dados provenientes de motores diferentes. Nos testes subsequentes, foi realizado *fine-tuning* nos modelos para adaptar seu desempenho a novos dados.



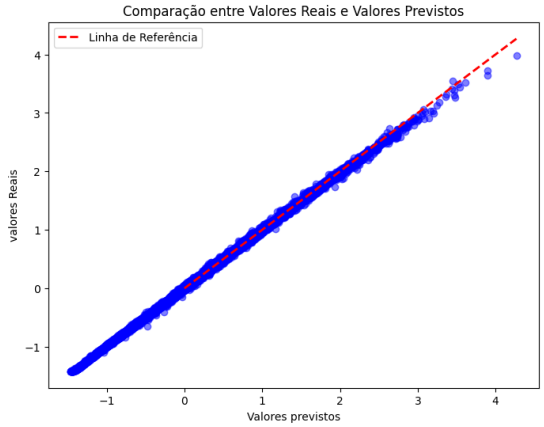
(a) 2D para V

(b) 2D para Nabla

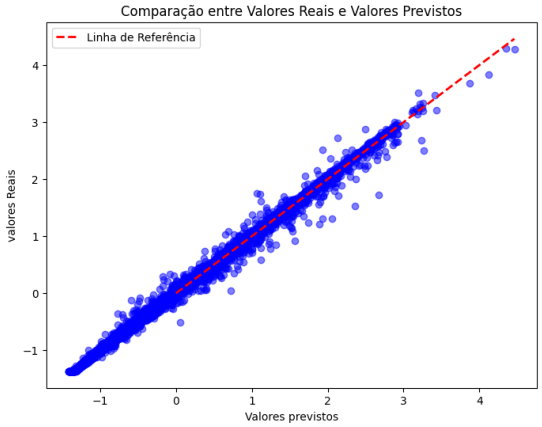
Figura 4.10 – Transfer-learning com camadas congeladas. (Target: hysteresis)

Na tentativa de preservar os parâmetros das camadas previamente treinadas no *"dataset"* original, o modelo foi treinado nos outros conjuntos de dados (V e Nabla), mantendo as camadas iniciais congeladas e ajustando apenas as camadas finais.(WANG et al., 2025)

A Figura 4.10 demonstra uma melhoria nas previsões do modelo 2D em comparação com as Figuras 4.7, 4.85 e 4.96. Contudo, o desempenho ainda está aquém da precisão esperada.



(a) 2D para V



(b) 2D para Nabla

Figura 4.11 – *Transfer-learning* re-treinando o modelo por completo. (*Target: hysteresis*)

Na Tabela 4.11 estão os resultados do *fine tuning* completo da rede 2D para V. Quando a rede é integralmente treinada, seu desempenho se mostra condizente com as expectativas de um modelo especializado.

Os experimentos demonstraram que, apesar da aplicação de técnicas de *Transfer Learning*, os modelos inicialmente treinados não conseguiram generalizar de forma satisfatória para conjuntos de dados de motores diferentes sem ajustes adicionais. No entanto, ao realizar o *fine-tuning* com camadas congeladas, foi possível observar alguma adaptação aos novos dados. Ainda assim, para que os modelos alcancem um desempenho comparável ao obtido em seus conjuntos de treinamento originais, foi necessário um re-treino completo, evidenciando que a transferência de aprendizado exige ajustes significativos para garantir resultados satisfatórios.

4.5 *Auto Encoders*

```
class Autoencoder(nn.Module):
    def __init__(self, input_dim, latent_dim):
        super(Autoencoder, self).__init__()
        # Encoder
        self.encoder = nn.Sequential(
            nn.Linear(input_dim, 64),
            nn.ReLU(),
            nn.Linear(64, latent_dim),
            nn.ReLU()
        )
        # Decoder
        self.decoder = nn.Sequential(
            nn.Linear(latent_dim, 64),
            nn.ReLU(),
```

```
15              nn.Linear(64, input_dim)
16          )
17
18      def forward(self, x):
19          latent = self.encoder(x)
20          reconstructed = self.decoder(latent)
21          return reconstructed, latent
```

Código 4.1 – Código do Autoencoder

O *decoder* tem como objetivo reconstruir a entrada original a partir da representação latente aprendida. Sua estrutura é semelhante à do *encoder*, expandindo a dimensão `latent_dim` para 64 neurônios e, posteriormente, para `input_dim`. Durante o treinamento, o modelo busca minimizar a diferença entre os dados originais e sua reconstrução, geralmente por meio de uma função de perda como o Erro Quadrático Médio (*Mean Squared Error*).

O método `forward` define o fluxo da informação na rede: a entrada é processada pelo *encoder*, gerando a variável `latent`, que em seguida é utilizada pelo *decoder* para produzir a reconstrução. O modelo retorna tanto a reconstrução quanto a representação latente, permitindo sua aplicação em tarefas como redução de dimensionalidade, extração de características e detecção de anomalias.

Este modelo *Encoder-Decoder* foi empregado para treinar a rede *autoencoder* utilizando os conjuntos de dados originais dos motores.

Após o treinamento, novas representações de entrada foram geradas, possibilitando a criação de conjuntos de dados artificiais. Esses dados foram consolidados em uma única base, reunindo informações de todos os motores analisados.

Em seguida, desenvolveu-se um modelo geral unificado, utilizando o maior modelo encontrado na seção de Auto ML, composto por 14 camadas com 61 neurônios cada e função de ativação *ReLU*, para realizar predições sobre os dados de teste originais.

Target	***Test MSE***	***Test MAE***	***Test MAPE***
2D Hysteresis	0.01752	0.01857	8.12
Nabla Hysteresis	0.00199	0.02713	54.86
V Hysteresis	0.00082	0.021509	14.72

Tabela 4.14 – Valores em *Hysteresis* para os modelos dos motores 2D, Nabla e V

Target	**Test MSE**	**Test MAE**	**Test MAPE**
2d Joule	0.18908	0.02916	15.74
Nabla Joule	0.00480	0.03496	23.76
V Joule	0.00141	0.02306	21.82

Tabela 4.15 – Valores em "Joule"para os modelos dos motores 2D, Nabla e V

Target	Test MSE	Test MAE	Test MAPE
2d Total	0.10414	0.02905	14.06
Nabla Total	0.00459	0.03975	61.70
V Total	0.00153	0.02617	21.16

Tabela 4.16 – Valores em "Total"para os modelos dos motores 2D, Nabla e V

As Tabelas 4.14, 4.15 e 4.16 apresentam os resultados dos modelos para cada *target*. Nota-se que esses resultados são competitivos em relação aos modelos específicos, evidenciando uma precisão similar à obtida individualmente por cada modelo especializado.

Ao final deste experimento, ficou demonstrada a viabilidade de desenvolver um modelo unificado capaz de integrar dados de múltiplas fontes, pois o desempenho alcançado foi comparável ao do modelo *XGBoost*.

4.6 Modelos Federados Únicos

Modelo	Test MSE	Test MAE	Test MAPE (%)
Centralizado 2D Hysteresis	0.0173	0.0139	6.29
Federado 2D Hysteresis	0.0172	0.0155	6.65

Tabela 4.17 – Comparação dos resultados do modelo 2D sobre “Hysteresis” em abordagem centralizada e federada

Modelo	Test MSE	Test MAE	Test MAPE (%)
Centralizado 2D Joule	0.1886	0.0206	10.34
Federado 2D Joule	0.1887	0.0234	10.34

Tabela 4.18 – Comparação dos resultados do modelo 2D sobre “Joule” em abordagem centralizada e federada

Modelo	Test MSE	Test MAE	Test MAPE (%)
Centralizado 2D Total	0.1033	0.0201	8.67
Federado 2D Total	0.1033	0.0207	8.45

Tabela 4.19 – Comparação dos resultados do modelo 2D sobre “Total” em abordagem centralizada e federada

As Tabelas 4.17, 4.18 e 4.19 mostram os resultados obtidos por um modelo treinado na base 2D. Em comparação com a abordagem centralizada, há pequenas diferenças de desempenho. Além disso, como as métricas permaneceram superiores aos valores de referência, concluiu-se que o modelo atingiu um nível de desempenho satisfatório, permitindo o avanço para os próximos experimentos.

A arquitetura utilizada foi a que apresentou melhor desempenho na etapa de *AutoML*, composta por 23 camadas ocultas, com 63 neurônios por camada, *learning rate* de 0.00016, 10 clientes federados, 15 épocas de treinamento por rodada e estratégia de agregação baseada em *weighted average*.

Modelo	**Test MSE**	**Test MAE**	**Test MAPE** (%)
Centralizado Nabla Hysteresis	0.0017	0.0246	34.13
Federado Nabla Hysteresis	0.0005	0.0169	12.27

Tabela 4.20 – Comparação dos resultados do modelo Nabla Hysteresis em abordagem centralizada e federada

Modelo	**Test MSE**	**Test MAE**	**Test MAPE** (%)
Centralizado Nabla Joule	0.0071	0.0488	31.62
Federado Nabla Joule	0.0068	0.0392	25.036

Tabela 4.21 – Comparação dos resultados do modelo Nabla "Joule"em abordagem centralizada e federada

Modelo	**Test MSE**	**Test MAE**	**Test MAPE** (%)
Centralizado Nabla Total	0.0031	0.0302	28.25
Federado Nabla Total	0.0033	0.0285	27.11

Tabela 4.22 – Comparação dos resultados do modelo Nabla "Total"em abordagem centralizada e federada

Nas Tabelas 4.20, 4.21 e 4.22, são apresentadas as métricas de um modelo especializado na base de dados Nabla, implementado em um ambiente federado. A análise dos resultados mostra que houve uma pequena diferença, evidenciando que a utilização do ambiente federado não impacta severamente o desempenho do modelo

A arquitetura utilizada foi a que apresentou melhor desempenho na etapa de *AutoML*, composta por 9 camadas ocultas, com 28 neurônios por camada, *learning rate* de 0.00014, 10 clientes federados, 15 épocas de treinamento por rodada e estratégia de agregação baseada em *weighted average*.

Modelo	**Test MSE**	**Test MAE**	**Test MAPE** (%)
Centralizado V Hysteresis	0.0003	0.0122	8.22
Federado V Hysteresis	0.0006	0.0173	16.02

Tabela 4.23 – Comparação dos resultados do modelo V Hysteresis em abordagem centralizada e federada

Modelo	Test MSE	Test MAE	Test MAPE (%)
Centralizado V Joule	0.0005	0.0130	9.49
Federado V Joule	0.0005	0.0129	9.45

Tabela 4.24 – Comparação dos resultados do modelo V "Joule"em abordagem centralizada e federada

Modelo	Test MSE	Test MAE	Test MAPE (%)
Centralizado V Total	0.0004	0.0130	12.95
Federado V Total	0.0004	0.0139	9.64

Tabela 4.25 – Comparação dos resultados do modelo V "Total"em abordagem centralizada e federada

Nas Tabelas 4.23, 4.24 e 4.25, são apresentadas as métricas de um modelo especializado para a base de dados V, implementado em um ambiente federado. A análise dos resultados indica que não foram observadas grandes variações, evidenciando que a utilização do ambiente federado afeta pouco o desempenho do modelo.

A arquitetura utilizada foi a que apresentou melhor desempenho na etapa de *AutoML*, composta por 18 camadas ocultas, com 61 neurônios por camada, *learning rate* de 0.00017, 10 clientes federados, 15 épocas de treinamento por rodada e estratégia de agregação baseada em *weighted average*.

Ao comparar os resultados dos modelos federados com os desenvolvidos na seção de *AutoML*, observa-se uma leve alteração de desempenho na abordagem federada. Porem, como os resultados ainda permanecem consistente em relação ao modelo *XGBoost*, foi considerado que a performance é suficientemente precisa para os próximos testes.

4.7 Modelo Federado Geral

Diferentemente do procedimento anterior, as bases não foram consolidadas; o modelo federado recebe, a cada rodada, dados aleatórios dos motores dos clientes, produzindo um efeito equivalente ao de uma base unificada dos três motores ao final.

Target	Test MSE	Test MAE	Test MAPE
2D Histeresis	0.4690	0.5661	288.90
2D Joule	0.7530	0.7394	264.06
2D Total	0.5294	0.6128	295.29

Tabela 4.26 – Resultado exemplo do modelo geral

Conforme demonstrado na Tabela 4.26, as estimativas do modelo geral apresentaram precisão significativamente inferior à do modelo *XGBoost* em todos os conjuntos de teste. As

especificações iniciais da simulação foram mantidas conforme descritas na seção "Modelos Federados Únicos". Mesmo com ajustes nos hiperparâmetros, como quantidade de clientes, tamanho da rede, *learning rate* e número de rodadas de treinamento, não foram observadas melhorias significativas nas métricas avaliadas. Isso evidencia a necessidade de uma investigação mais aprofundada dos métodos de agregação e da forma como os dados estão sendo codificados.

5 Considerações Finais

O objetivo final do projeto foi parcialmente alcançado. Constatou-se a existência de um modelo capaz de generalizar para múltiplas bases de dados com diferentes números de entradas, conforme demonstrado na seção de Auto Encoders.

Contudo, os experimentos não se comportaram conforme o esperado, e não foi possível implementar esse modelo geral em um ambiente federado simulado.

5.1 Conclusão

Os resultados deste projeto demonstraram a viabilidade de utilizar redes neurais compostas exclusivamente por camadas totalmente conectadas para a realização de predições robustas. Adicionalmente, as técnicas de *Transfer Learning* e *Auto-Encoders* permitiram a criação de modelos capazes de generalizar para múltiplas bases de dados.

Destaca-se que, embora o *XGBoost* continue apresentando desempenho competitivo, a abordagem baseada em redes neurais evidenciou uma melhoria expressiva no alvo Total, evidenciando sua maior flexibilidade ao lidar com predições que envolvem múltiplos valores.

Os resultados obtidos utilizando Aprendizado Federado foram ligeiramente inferiores aos alcançados no ambiente centralizado. Essa diferença pode estar relacionada à definição dos hiperparâmetros e à arquitetura do modelo, pois foi utilizado um único modelo para realizar as predições de todos os alvos para cada motor. Consequentemente, a arquitetura não é especializada para cada alvo, ao contrário dos modelos desenvolvidos na seção "Auto ML". No entanto, os resultados foram bastante próximos, sugerindo que há potencial para melhorias nas métricas mediante uma investigação mais aprofundada dos hiperparâmetros e das arquiteturas empregadas na federação.

5.2 Trabalhos Futuros

Investigações mais aprofundadas sobre o ambiente e estratégias de agregação são necessárias para confirmar a viabilidade de desenvolver um modelo tão robusto quanto o centralizado, mas em uma arquitetura descentralizada.

1. Autor. Título do Artigo. Cidade: Conferência, Ano.

Referências

ABRAHAM, T. H. (physio)logical circuits: The intellectual origins of the mccullochpitts neural networks. *Journal of the History of the Behavioral Sciences*, Wiley, v. 38, n. 1, p. 3–25, 2002. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/jhbs.1094>.

AL-GABALAWY, M.; ELMETWALY, A. H.; YOUNIS, R. A.; OMAR, A. I. Temperature prediction for electric vehicles of permanent magnet synchronous motor using robust machine learning tools. *Journal of Ambient Intelligence and Humanized Computing*, 2022.

BENGIO, Y.; COURVILLE, A.; VINCENT, P. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, v. 35, n. 8, p. 1798–1828, 2013. Disponível em: <https://arxiv.org/abs/1206.5538>.

BOLDEA, I.; TUTELEA, L. N.; PARSA, L.; DORRELL, D. Automotive electric propulsion systems with reduced or no permanent magnets: An overview. *IEEE Transactions on Industrial Electronics*, v. 61, n. 10, p. 5696–5711, 2014.

BOTCHKAREV, A. A new typology design of performance metrics to measure errors in machine learning regression algorithms. *Interdisciplinary Journal of Information, Knowledge, and Management*, Informing Science Institute, v. 14, p. 045076, 2019. ISSN 1555-1237. Disponível em: <http://dx.doi.org/10.28945/4184>.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.

HERNANDEZ, R. R.; BAKER, K.; BARRETTO, M.; JARAMILLO, P.; SAMARAS, C. Public health and climate benefits of electric vehicles. *Environmental Research Letters*, v. 15, n. 9, p. 094056, 2020. Disponível em: <https://iopscience.iop.org/article/10.1088/1748-9326/ab9c8e>.

HINTON, G. E.; SALAKHUTDINOV, R. R. Reducing the dimensionality of data with neural networks. *Science*, American Association for the Advancement of Science, v. 313, n. 5786, p. 504–507, 2006.

JORDAN, M. I.; MITCHELL, T. M. Machine learning: Trends, perspectives, and prospects. *Science*, v. 349, n. 6245, p. 255–260, jul. 2015.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, v. 5, n. 4, p. 115–133, 1943.

MCMAHAN, H. B.; MOORE, E.; RAMAGE, D.; ARCAS, B. A. y. Communication-efficient learning of deep networks from decentralized data. *CoRR*, abs/1602.05629, 2016. Disponível em: <http://arxiv.org/abs/1602.05629>.

MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. Efficient estimation of word representations in vector space. In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. [s.n.], 2013. Disponível em: <http://arxiv.org/abs/1301.3781>.

NIELSEN, M. A. *Neural Networks and Deep Learning*. Determination Press, 2015. Disponível em: <http://neuralnetworksanddeeplearning.com/>.

NVIDIA. *Introduction to Federated Learning*. 2024. <https://nvflare.readthedocs.io/en/2.6/fl_introduction.html>. Accessed: 2025-08-02.

PAN, S. J.; YANG, Q. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, v. 22, p. 1345–1359, 2010. Disponível em: <https://api.semanticscholar.org/CorpusID:740063>.

PARMEZAN, A.; AQUINO, A. L.; OLIVEIRA, L. S.; SABOURIN, R. Evaluation of feature selection methods for ensemble-based one-class classification. *Information Sciences*, Elsevier, v. 496, p. 318–333, 2019. Disponível em: <https://doi.org/10.1016/j.ins.2019.05.054>.

SHEN, Y.; MA, T.; LI, J. Investigation of dynamic fractures under varying stress states. *International Journal of Mechanical Sciences*, v. 293, p. 110177, 2025. ISSN 0020-7403. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0020740325002632>.

SHIMIZU, Y. Efficiency optimization design that considers control of interior permanent magnet synchronous motors based on machine learning for automotive application. *IEEE Access*, v. 11, p. 41–49, 2023.

SHIN, Y.-M.; KOO, D.-H.; SHIN, M.-H.; PARK, M.-H. Pm design of ipmsm using parameterized finite element model. In: *2014 International Conference on Electrical Machines and Systems (ICEMS)*. IEEE, 2014. p. 618–621. Disponível em: <https://www.researchgate.net/publication/273594716_PM_Design_of_IPMSM_using_Parameterized_Finite_Element_Model>.

SIMEONE, O. A very brief introduction to machine learning with applications to communication systems. *IEEE Journal on Selected Areas in Communications*, v. 37, n. 6, p. 1282–1295, 2018. Disponível em: <https://arxiv.org/abs/1808.02342>.

WANG, H.; RAJ, B. On the origin of deep learning. *arXiv preprint arXiv:1702.07800*, 2017. Disponível em: <https://arxiv.org/pdf/1702.07800>.

WANG, H.; WANG, J.; ZHAO, Z.; TAN, Y.; WU, Y.; LIU, H.; YANG, J.; ZHANG, E.; CHEN, X.; RONG, Z.; GUO, S.; LI, Y. *Understanding Knowledge Transferability for Transfer Learning: A Survey*. 2025. Disponível em: <https://arxiv.org/abs/2507.03175>.

YADAV, N.; YADAV, A.; KUMAR, M. History of neural networks. In: *An Introduction to Neural Network Methods for Differential Equations*. Springer, 2015. p. 1–14. Disponível em: <https://link.springer.com/chapter/10.1007/978-94-017-9816-7_2>.

YOU, Y.-m. Multi-objective optimal design of permanent magnet synchronous motor for electric vehicle based on deep learning. *Applied Sciences*, v. 10, n. 2, p. 482, 2020.

ZHANG, C.; XIE, Y.; BAI, H.; YU, B.; LI, W.; GAO, Y. A survey on federated learning. *Knowledge-Based Systems*, v. 216, p. 106775, 2021.