UNIVERSIDADE FEDERAL DO PARÁ

INSTITUTO DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Wilson Antonio Cosmo Macedo

Estimação de Descarga de Dispositivo IoT usando Deep Learning com Otimização NSGA-II

DM 06/2024

Belém

2024

Wilson Antonio Cosmo Macedo

Estimação de Descarga de Dispositivo IoT usando Deep Learning com Otimização NSGA-II

DM 06/2024

Dissertação submetida à Banca Examinadora do Programa de Pós-Graduação em Engenharia Elétrica da UFPA para obtenção do Grau de Mestre em Engenharia Elétrica na Área de Telecomunicações.

Universidade Federal do Pará

Orientador: Dr. Fabrício José Brito Barros

Belém

2024

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD
Sistema de Bibliotecas da Universidade Federal do Pará
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)

M141e Macedo, Wilson Antonio Cosmo.
Estimação de Descarga de Dispositivo IoT usando Deep Learning com Otimização NSGA-II / Wilson Antonio Cosmo Macedo. — 2024.
XVI, 78 f. : il. color.

Orientador(a): Prof. Dr. Fabrício José Brito Barros
Dissertação (Mestrado) - Universidade Federal do Pará, Instituto de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, Belém, 2024.

1. IoT. 2. LoRa. 3. Descarga. 4. Deep Learning. 5. NSGA-II. I. Título.

CDD 384

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

“ESTIMAÇÃO DE DESCARGA DE DISPOSITIVO IOT USANDO DEEP LEARNING COM OTIMIZAÇÃO NSGA-II”

AUTOR: **WILSON ANTÔNIO COSMO MACÊDO**

DISSERTAÇÃO DE MESTRADO SUBMETIDA À BANCA EXAMINADORA APROVADA PELO COLEGIADO DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA, SENDO JULGADA ADEQUADA PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA ELÉTRICA NA ÁREA DE TELECOMUNICAÇÕES.

APROVADA EM: 28/02/2024

BANCA EXAMINADORA:

Prof. Dr. Fabrício José Brito Barros
(Orientador – PPGEE/ITEC/UFPA)

Prof.ª Dr.ª Jasmine Priscyla Leite de Araújo
(Avaliadora Interna – PPGEE/ITEC/UFPA)

Prof. Dr. Rommel Thiago Jucá Ramos
(Avaliador Externo ao Programa – ICB/UFPA)

Prof. Dr. João Victor Costa Carmona
(Avaliador Externo – UNIFESSPA)

VISTO:

Prof. Dr. Diego Lisboa Cardoso

(Coordenador do PPGEE/ITEC/UFPA)

Dedico este trabalho aos meus pais, irmã e a todos que acreditam verdadeiramente na importância do conhecimento e sua aplicação em benefício da sociedade.

Agradecimentos

Agradeço especialmente à minha família, meus pais, Marta Cosmo e Wilson Macêdo, e minha irmã Flávia Cosmo, pela confiança e apoio que sempre depositaram em mim.

Ao meu orientador, Professor Fabrício Barros, pela orientação fornecida a mim e pelos seus esforços em prol do desenvolvimento da Pesquisa Científica Brasileira, bem como à todos os professores do Programa com os quais eu tive a oportunidade de adquirir valiosos conhecimentos.

Aos meus amigos, que de algum modo tentaram me auxiliar ou apoiar de forma sincera, e aos meus colegas de grupo de pesquisa que comigo se esforçam para demonstrar a importância da ciência e de sua aplicação em benefício à sociedade.

"Ninguém pode construir em teu lugar as pontes que precisarás para atravessar o rio da vida – ninguém, exceto tu, só tu."

Friedrich Nietzsche, 1874

Resumo

O aumento das aplicações de redes IoT (Internet das Coisas) destaca a necessidade de otimizar a gestão de energia nestes sistemas, pois a eficiência energética é crucial para a adaptabilidade das implementações que referem-se à IoT. Este estudo analisa as curvas de descarga de uma bateria recarregável em um contexto de rede IoT que utiliza comunicação LoRa (Long Range) e vários sensores, com o objetivo de gerar múltiplas curvas de descarga para estimar o comportamento da bateria nesse cenário. Essas curvas foram utilizadas para treinar uma Rede Neural Artificial (RNA) de várias camadas, implementando técnicas de Deep Learning, na qual a arquitetura da RNA foi delineada usando o algoritmo de Otimização Multiobjetivo NSGA-II (Non-dominated Sorting Genetic Algorithm II), o que resultou em modelos com capacidade de estimar o tempo de descarga da bateria ao analisar um segmento do processo de descarga observado pelo modelo com erro médio quadrático de aproximadamente dois minutos para o modelo mais eficiente encontrado. Este resultado representa uma margem muito positiva, visto que a extensão dos testes de descarga são de até aproximadamente setenta e uma horas e a taxa de amostragem de coleta dos dados é de um minuto.

Palavras-chave: IoT, LoRa, Descarga, Deep Learning, NSGA-II.

Abstract

The increasing adoption of IoT (Internet of Things) network applications highlights the need to optimize energy management in these systems, because energy efficiency is crucial for the adaptability of IoT implementations. This study analyzes the discharge curves of a rechargeable battery in an IoT network context utilizing LoRa (Long Range) communication and various sensors, with the objective of generating multiple discharge curves to estimate the battery behavior in this scenario. These curves were used to train a Multilayer Artificial Neural Network (ANN), implementing Deep Learning techniques, where the ANN architecture was outlined using the NSGA-II (Non-dominated Sorting Genetic Algorithm II) Multi-objective Optimization algorithm. This resulted in models capable of estimating the battery discharge time by analyzing a segment of the discharge process observed by the model with a mean squared error of approximately two minutes for the most efficient model found. This result represents a very positive margin, considering that the duration of the discharge tests extends to approximately seventy-one hours and the data collection sampling rate is one minute.

Keywords: IoT, LoRa, Discharge, Deep Learning, NSGA-II

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AI	Inteligência Artificial
AIoT	Inteligência Artificial na Internet das Coisas
CAGR	Taxa de Crescimento Anual Composta
CNN	Convolutional Neural Network
CSS	Chirp Spread Spectrum
DCNN	Rede Neural Convolucional Profunda
EBa	Ensemble Bagging
EBo	Ensemble Boosting
EE	Eficiência Energética
ESP32	Espressif Systems Platform 32
GPR	Regressão por Processo Gaussiano
IoT	Internet das Coisas
Li-Ion	Íon de Lítio
LoRa	Long Range
LoRaWAN	Long Range Wide Area Network
LR	Regressão Linear
LSTM	Long Short-Term Memory
LPWAN	Low-Power Wide-Area Network
MCU	Microcontroller Unit
MLP	Perceptron de Múltiplas Camadas
MSE	Erro Médio Quadrático
NB-IoT	Narrowband Internet of Things
NSGA-II	Non-dominated Sorting Genetic Algorithm II

NSF	Fator de Espalhamento
RNN	Redes Neurais Recorrentes
RSSF	Redes de Sensores Sem Fio
SF	Fator de Espalhamento
SNR	Razão Sinal-Ruído
SoC	Estado de Carga
SVM	Máquina de Vetor de Suporte
WiFi	Wireless Fidelity

Sumário

1 Introdução

A ascensão das redes de sensores IoT (Internet das Coisas) tem revolucionado a forma como interagimos e coletamos dados em ambientes diversos, e suas aplicações estão cada vez mais presentes nos mais diversos contextos. Segundo (MONTEIRO, 2023) entre 2022 e 2027, a venda de dispositivos IoT deve ter uma taxa de crescimento anual composta (CAGR) de 19,8%, e com isso o País vai passar de 300 milhões de dispositivos vendidos em 2022 para cerca de 700 milhões ao final de cinco anos. Essas redes, compostas por dispositivos sensores distribuídos, desempenham um papel crucial na obtenção de informações em tempo real, possibilitando a tomada de decisões mais eficientes.

No entanto, um desafio significativo que acompanha o avanço dessas redes é o gerenciamento da energia dos dispositivos, especialmente no que diz respeito às baterias que alimentam esses sensores. O consumo de energia em dispositivos IoT, muitas vezes alimentados por baterias autônomas, é uma preocupação central devido às limitações inerentes ao tamanho e capacidade dessas fontes de energia. Dada essa necessidade, de acordo com (ABINC, 2023) para 2024 espera-se um foco crescente em soluções IoT que contribuam para práticas sustentáveis, como a eficiência energética, o monitoramento ambiental e a redução do desperdício de recursos.

Um aspecto importante para elaborar soluções que visam a gerência energética no IoT é observar as tendências adotadas nas redes LPWAN (*Low-Power Wide-Area Network*). Segundo (ZABEU, 2023), NB-IoT e LoRaWAN representarão 87% de todas as conexões LPWAN em 2028, sendo essas duas tecnologias sem fio as que operam com maior eficiência energética, portanto, é interessante levar em consideração essas tecnologias sem fio devido o seu comportamento de baixo consumo de energia e sua alta aplicação no mercado LPWAN, especialmente a tecnologia LoRa, pois esta pode ser implementada com frequências livres de registro.

Outro aspecto importante nas aplicações IoT são a presença de Inteligência Artificial (IA) no contexto da Internet das Coisas, o que é conhecido com $AIoT$, e conforme (ASCENTY, 2023) o mercado global de $AIoT$ vai crescer 39,1% até 2027, alcançando o US$ 83,6 bilhões, o que explicita uma realidade na qual essas aplicações já estão bastante difundidas e com grande tendência de crescimento futuro, o que torna propício o surgimento de novas formas de aplicar os algoritmos de aprendizado nesse contexto para além das abordagens de análise e classificação de dados coletados, estendendo a aplicação das IAs para aplicações de avaliação e gerência da própria rede de forma inteligente.

1.1 Motivação e Justificativa

A motivação intrínseca para o desenvolvimento desta pesquisa, emerge da necessidade premente de endereçar um obstáculo crucial que permeia a eficiência e a viabilidade das redes de sensores IoT, que é o gerenciamento energético. À medida que testemunhamos uma proliferação exponencial dessas redes em nosso cotidiano, desde sistemas urbanos inteligentes até ambientes industriais e aplicações agrícolas, torna-se imperativo confrontar os desafios associados à autonomia e longevidade das baterias que alimentam esses dispositivos distribuídos.

O cerne da questão reside na dualidade entre a crescente demanda por dados em tempo real e as limitações inerentes ao armazenamento e fornecimento de energia em dispositivos IoT. Baterias autônomas, frequentemente utilizadas nesses sensores, apresentam uma capacidade limitada, levando a uma constante preocupação com a gestão eficaz da energia para garantir a operacionalidade contínua dos dispositivos, e a crescente aplicação da IA no contexto IoT torna propício o surgimento de novas propostas, como o modelo de previsão generalista do presente trabalho.

Ao abordar essa problemática, visa-se não apenas superar os desafios técnicos associados à previsão do tempo de descarga das baterias, mas também contribuir para a consolidação de uma infraestrutura IoT mais robusta e sustentável. A capacidade de antecipar com precisão o tempo restante de operação, com um modelo generalista, não só otimiza a utilização dos recursos energéticos disponíveis, mas também oferece uma base sólida para a implementação de estratégias proativas de manutenção e substituição de baterias.

Além disso, considerando o impacto ambiental do descarte inadequado de baterias e a busca por soluções mais ecológicas, o desenvolvimento de métodos eficazes de gerenciamento de energia para dispositivos IoT assume uma importância ainda maior. Esta pesquisa, portanto, é motivada por uma visão de promover a eficiência operacional, a sustentabilidade e a expansão contínua do alcance e impacto das redes de sensores IoT em nossa sociedade cada vez mais conectada.

1.2 Objetivos

O objetivo geral deste projeto visa desenvolver um modelo de estimação de tempo de descarga de bateria, baseado somente na observação da curva de descarga gerada pelo dispositivo IoT, implementado a partir de um cenário proposto, tornando o modelo generalista, devido ao fato do mesmo não necessitar de informações específicas sobre o dispositivo ou os sensores conectados ao mesmo, pois a estimação se dá a partir da observação de uma fração do próprio processo de descarga, que é referido neste trabalho

como janela de observação, criando uma estimativa da vida útil restante do dispositivo com a margem de erro de alguns poucos minutos, utilizando técnicas de *Deep Learning* e otimização multiobjetivo, para alçar a menor margem de erro com a menor janela de observação. Para alçar o presente objetivo geral, o trabalho atingiu os seguintes objetivos específicos:

- Definição do Cenário Proposto, identificando e caracterizando o cenário de implantação do dispositivo IoT, considerando os sensores presentes no Nó final.
- Desenvolvimento da Infraestrutura de Coleta de Dados, implementando um sistema de coleta de dados para registrar diferentes curvas de descarga do dispositivo IoT a partir de diferentes combinações de sensores.
- Desenvolver uma arquitetura de rede neural, capaz de analisar a curva de descarga e estimar o tempo de vida útil restante do dispositivo, levando em consideração as características dos dados obtidos e a janela de Observação.
- Incorporar técnicas de otimização multiobjetivo, para equilibrar a minimização da margem de erro de estimação com a redução da janela de observação, buscando encontrar a topologia mais adequada do modelo definido para atingir esses parâmetros.
- Validação dos resultados, que ocorre durante o processo de otimização uma vez que cada modelo candidato é avaliado com dados de um conjunto de dados separado dos que foram utilizados para o treinamento dos modelos.

1.3 Estrutura do Trabalho

O presente trabalho segue uma estrutura sequencial, organizada para proporcionar uma compreensão abrangente do tema abordado. O Capítulo 2, apresenta diversos trabalhos correlatos, oferecendo uma revisão da literatura relacionada ao tema. Isso inclui uma descrição e um comparativo entre diferentes abordagens, estabelecendo um contexto para as contribuições do trabalho.

No Capítulo 3, o referencial teórico explora conceitos-chave necessários para compreender o desenvolvimento do projeto, como redes de sensores, Internet das Coisas, redes LPWAN, redes neurais artificiais e *Deep Learning*.

O Capítulo 4, detalha a metodologia adotada, descrevendo o cenário proposto, a prototipagem, a coleta de dados e o treinamento do modelo estimador de descarga. Essa seção oferece detalhes sobre como os objetivos propostos foram abordados e implementados.

Os resultados obtidos são apresentados no Capítulo 5, com uma análise do processo de otimização e uma avaliação dos modelos encontrados. Este capítulo destaca as soluções

derivadas dos modelos encontrados, bem como uma análise da qualidade dessas soluções de acordo com a métrica adotada.

A conclusão, no Capítulo 6, recapitula os principais pontos discutidos, apresentando também sugestões para trabalhos futuros. O trabalho é encerrado com a lista de referências, destacando as fontes utilizadas para embasar a pesquisa realizada.

2 Trabalhos correlatos

2.1 Descrição

No artigo de (SHEN, 2019) é abordado o crescente uso de baterias recarregáveis de íon de lítio (Li-Ion) nas últimas duas décadas e destaca a importância dos sistemas de gerenciamento de bateria para monitorar o estado de saúde em tempo real. Introduz um método de aprendizado profundo, utilizando uma rede neural convolucional profunda (DCNN), para estimar a capacidade de células Li-Ion com base em medições de voltagem, corrente e capacidade de carga durante ciclos de carga parciais. O estudo, uma das primeiras tentativas de aplicar aprendizado profundo à estimativa online de capacidade de baterias Li-Ion, utiliza dados de ciclagem de células para validar o desempenho do método proposto. Comparado a métodos tradicionais de aprendizado de máquina, como redes neurais rasas e máquina de vetor de relevância (RVM), o método de aprendizado profundo demonstra maior precisão e robustez na estimativa online da capacidade de bateria Li-Ion.

Já a obra de (LI; YANG; WANG, 2020) destaca o potencial do LoRa para conectar dispositivos IoT em larga escala, enfatizando sua capacidade de proporcionar comunicação de longo alcance com baixo consumo de energia. Propõe o DyLoRa, um sistema dinâmico de controle de transmissão LoRa, visando melhorar a eficiência energética ao ajustar os parâmetros conforme diferentes ambientes. O estudo baseia-se na relação entre a taxa de erro de símbolos e a Razão Sinal-Ruído (SNR), derivando modelos para caracterizar a eficiência energética e ajustar os parâmetros dinamicamente, e não são mencionadas análises específicas relacionadas a baterias, e a limitação do LoRa em termos de taxa de dados e dados esparsos é destacada como um desafio, enfatizando a importância do ajuste de parâmetros para otimizar a eficiência energética em ambientes com tráfego LoRa de baixa taxa.

O foco do artigo de (SU; QIN; NI, 2020) trata-se de uma técnica para redes de baixo consumo de energia, destacando a importância da eficiência energética (EE) em dispositivos LoRa alimentados por bateria. A pesquisa concentra-se na alocação eficiente de recursos para maximizar a EE do sistema (SEE) e a EE mínima (MEE) dos usuários LoRa. Propõe algoritmos subótimos e ótimos que abordam conjuntamente o agendamento de usuários, a atribuição de fator de espalhamento (SF) e a alocação de potência de transmissão, destacando-se a formulação abrangente dos problemas de otimização, a decomposição em subproblemas e a contribuição significativa para a EE em redes LoRa.

A obra de (NURGALIYEV, 2020) apresenta um método para prever a vida útil de uma rede de sensores sem fio baseada nos módulos sem fio LoRa Ra-01. Para desenvolver

um modelo de previsão do consumo de energia, módulos de sensores sem fio foram montados, e dados experimentais foram obtidos usando o ambiente de desenvolvimento LabView. Experimentos foram conduzidos para obter a curva de descarga da bateria, e dados experimentais sobre o consumo de energia em relação ao comprimento do pacote foram obtidos no modo de transmissão. Utilizando esses dados experimentais, foram estabelecidas relações entre a vida útil do sistema, a duração do modo de espera e o comprimento do pacote. O artigo também aborda uma abordagem probabilística para prever a vida útil do sistema com base na probabilidade de transmissão de dados durante o dia, utilizando cadeias de Markov.

O estudo de (CHANDRAN, 2021) aborda a estimativa do estado de carga (SoC) de sistemas de bateria de íon de lítio em veículos elétricos, reconhecendo a complexidade da durabilidade e confiabilidade dos sistemas de gerenciamento de bateria. Devido à não linearidade do processo de degradação da bateria, prever a estimativa do SoC com menos degradação é uma tarefa desafiadora. O estudo apresenta a estimativa do SoC usando seis algoritmos de aprendizado de máquina para aplicação em veículos elétricos: rede neural artificial (RNA), máquina de vetor de suporte (SVM), regressão linear (LR), regressão por processo gaussiano (GPR), ensemble bagging (EBa) e ensemble boosting (EBo).

O artigo de (TAMILSELVI, 2021) realiza um estudo abrangente de vários modelos de bateria, incluindo modelos eletroquímicos, matemáticos, orientados por circuito e modelos combinados para diferentes tipos de baterias. Além disso, discute as vantagens e desvantagens desses tipos de modelagem. O trabalho discute também a aplicação de vários algoritmos de aprendizado de máquina e heurística para sistemas de gerenciamento de bateria, e detalha as características de carga e descarga usando técnicas de caixa preta e caixa cinza para modelar a bateria de íon de lítio, analisando abordagens, vantagens e desvantagens desses tipos de modelagem.

A abordagem de (VÄÄNÄNEN; HÄMÄLÄINEN, 2021) avalia e mede a eficiência de algoritmos simples de compressão temporal na redução do consumo de energia de nós de sensores baseados em LoRa. Considerando que a transmissão de rádio é a operação mais consumidora de energia em um nó de sensor sem fio, três algoritmos leves de compressão são implementados em uma plataforma LoRa embarcada para comprimir dados do sensor em modo online. O consumo de energia é comparado com a situação sem a implementação de nenhum algoritmo de compressão. Os resultados indicam que um algoritmo simples de compressão é eficaz para melhorar a vida útil de um nó de sensor alimentado por bateria. Apesar do alto consumo de energia, durante a transmissão de rádio, o consumo de corrente em modo de espera é um fator significativo para a vida útil geral do dispositivo, especialmente em intervalos de medição longos.

O artigo de (RAJAB, 2023) aborda a demanda por nós de sensores em implementações de IoT, destacando a necessidade de operação confiável por períodos estendidos. A

ênfase é dada às configurações de rádio capazes de suportar uma transmissão de alta taxa de dados, otimizando a eficiência energética. A tecnologia LoRa/LoRaWAN é destacada como uma das principais em redes de área ampla e de baixa potência (LPWAN), sendo notável por sua eficiência energética. As limitações energéticas são consideradas um desafio significativo em redes de sensores sem fio, uma vez que a vida útil da bateria que alimenta os nós sensores é restrita e muitas vezes não recarregável.

2.2 Comparativo

A Tabela 1 compara as obras citadas de acordo com a aplicação específica a qual são direcionados, como se são voltados para análise de consumo energético do dispositivo, se estão analisando o comportamento da bateria, se estão utilizando dispositivos LoRa/LoRaWAN nas suas análises e se suas metodologias foram atingidas por meio de Modelos Matemáticos (MM) ou por Machine Learning (ML).

Tabela 1. Análise comparativa dos tópicos abordados pelos trabalhos correlatos.

Autor	Consumo	LoRa	Bateria	MM	ML
(SHEN, 2019)	Sim	Não	Sim	Não	Sim
(LI; YANG; WANG, 2020)	Sim	Sim	Não	Não	Não
(SU; QIN; NI, 2020)	Sim	Sim	Sim	Sim	Não
(NURGALIYEV, 2020)	Sim	Sim	Sim	Não	Não
(CHANDRAN, 2021)	Sim	Não	Sim	Sim	Sim
(TAMILSELVI, 2021)	Não	Não	Sim	Sim	Sim
(VÄÄNÄNEN; HÄMÄLÄINEN, 2021)	Sim	Sim	Não	Não	Não
(RAJAB, 2023)	Sim	Sim	Não	Não	Não

Em síntese, os trabalhos correlatos apresentados na Tabela 1 apresentam uma diversidade de abordagens relacionados aos tópicos pertinentes à presente pesquisa. (SHEN, 2019) contribuem com um estudo que considera o consumo de energia e a aplicação de Machine Learning para otimizar o desempenho. (LI; YANG; WANG, 2020) exploram aspectos do consumo de energia e a utilização da tecnologia LoRa, embora não tenham incorporado métodos de aprendizado profundo em sua análise.

(SU; QIN; NI, 2020) destacam-se ao abordar o consumo de energia, LoRa, bateria e a aplicação de machine learning, fornecendo uma abordagem mais abrangente. (NURGALIYEV, 2020) também consideram aspectos como consumo de energia e LoRa, embora não tenham adotado técnicas de aprendizado profundo em sua pesquisa. (CHANDRAN, 2021) apresentam uma contribuição significativa ao incorporar análise de consumo, utilização do LoRa, gestão de bateria e a aplicação de machine learning. (TAMILSELVI, 2021) se concentram na gestão de bateria, com destaque para aspectos como consumo de energia

e LoRa. (VÄÄNÄNEN; HÄMÄLÄINEN, 2021) abordam o consumo e LoRa, enquanto (RAJAB, 2023) exploram o consumo e a tecnologia LoRa, ambos sem a utilização de técnicas de aprendizado profundo.

Ao considerar essas contribuições, propõe-se integrar efetivamente os elementos abordados por esses autores, como consumo de energia, LoRa, gestão de bateria e deep learning, para desenvolver uma abordagem abrangente da estimação da descarga de bateria em dispositivos LoRa, com a adição de uma nova análise baseada somente na observação do processo de descarga, proporcionado por um modelo Deep Learning robustamente otimizado por um algoritmo de Otimização Multiobjetivo.

3 Referencial teórico

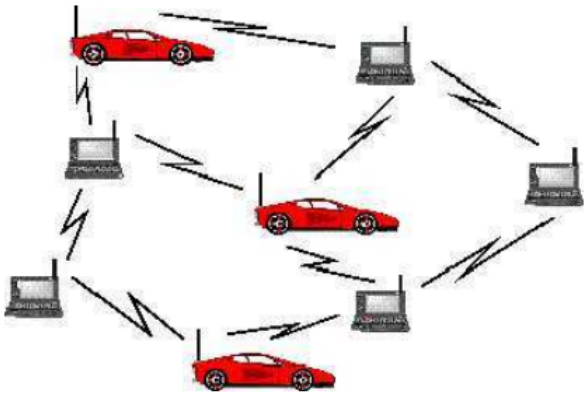
3.1 Redes de Sensores

Dado que um sensor é um dispositivo sensível a alguma forma de energia, que pode ser luminosa, térmica ou cinética (THOMAZINI; ALBUQUERQUE, 2020), as Redes de Sensores representam sistemas distribuídos compostos por uma coleção de sensores, os quais colaboram de maneira sinérgica para coletar, processar e transmitir dados relativos ao ambiente no qual estão incorporados. Conforme afirmado por (PEREIRA; AMORIM; CASTRO, 2003), a definição de redes de sensores, refere-se a um conjunto de nós individuais (sensores) que operam de forma independente, mas que podem estabelecer uma conexão em rede com o intuito de consolidar as informações individuais de cada sensor para monitorar algum fenômeno específico.

As Redes de Sensores Sem Fio (RSSF) são compostas por nodos sensoriais que se comunicam por meios como radiofrequência ou comunicação infravermelha, onde cada sensor individual é dotado de sensores específicos, processadores de baixo consumo e dispositivos de comunicação, possibilitando a coleta de dados, o processamento local de informações e a troca colaborativa de resultados entre os sensores na rede. Destaca-se que essas redes apresentam particularidades, como o uso de recursos energéticos restritos, topologia de rede dinâmica e uma elevada densidade de nós, conforme mencionado por (PEREIRA; AMORIM; CASTRO, 2003).

A topologia das RSSF pode variar de acordo com a aplicação específica, podendo ser configuradas em malha, estrela ou hierárquica. A escolha da topologia depende dos requisitos da aplicação e das características do ambiente em que estão implantadas, e segundo (LOUREIRO, 2003) podem ser vistas como um tipo especial de rede movel ad hoc, onde os elementos computacionais trocam dados diretamente entre si, como ilustrado na Figura 1.

Figura 1. Exemplo de rede ad hoc

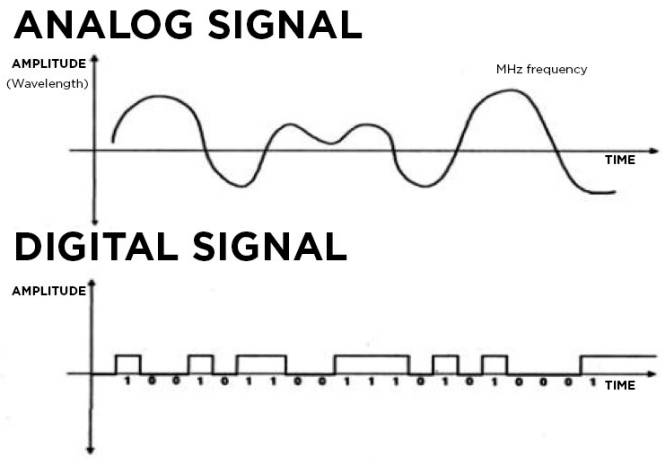


Fonte: (LOUREIRO, 2003)

3.1.1 Sensores

De acordo com (ANALOG DEVICES, INC, 2024) um sensor elétrico é um dispositivo que detecta um parâmetro físico de interesse, tais como calor, luz, som, e o converte em um sinal elétrico que pode ser medido e utilizado por um sistema elétrico ou eletrônico, permitindo o monitoramento de fenômenos físicos e a coleta de informações sobre o ambiente o qual o sensor se encontra. Segundo (JOSTJUN, 2019) existem dois tipos de sensores eletrônicos, os analógicos e digitais, onde os sensores analógicos convertem dados físicos em um sinal contínuo analógico, ao contrário dos digitais que estão limitados a um conjunto finito de valores possíveis, como explicitado na Figura 2.

Figura 2. Diferença de leitura dos sensores analógicos e digitais.



Fonte: (APOGEEWEB, 2018)

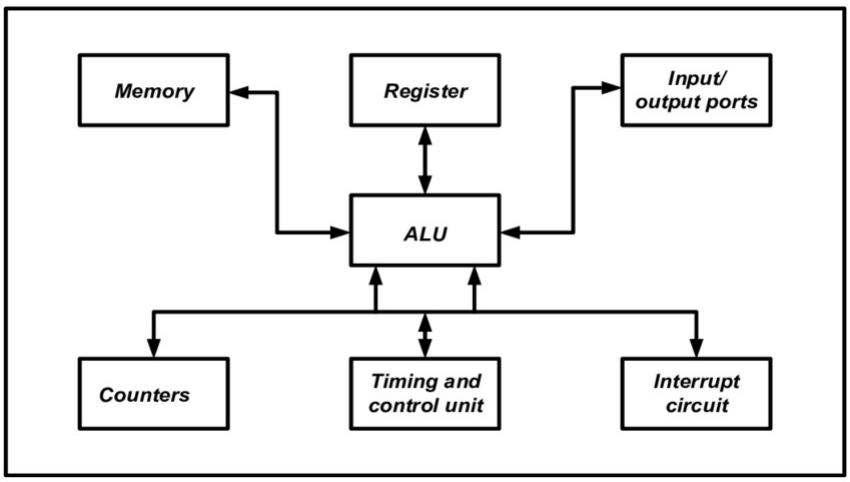
Visto que os sensores podem ser utilizados para monitorar ambientes que sejam de difícil acesso ou perigosos, tais como o fundo do oceano, vizinhanças de atividades vulcânicas, áreas de desastres e campos de atividade nuclear (PEREIRA; AMORIM; CASTRO, 2003), as aplicações das redes de sensores são diversas e abrangem áreas como monitoramento ambiental, agricultura de precisão, saúde, segurança e automação industrial.

3.1.2 Microcontroladores

Um microcontrolador é um computador de chip único fabricado especificamente para aplicações de controle embutido, sendo dispositivos de baixo custo que podem ser utilizados facilmente em aplicações de controle digital (IBRAHIM, 2006), sendo largamente aplicados em sistemas embarcados. Sua aplicação é vasta, abrangendo desde eletrodomésticos simples até sistemas complexos em automóveis, dispositivos médicos, equipamentos industriais e dispositivos IoT.

De acordo com digital (ELECTRONICSHUB, 2017) um microcontrolador é composto por uma Unidade Central de Processamento (CPU), Memória Somente para Leitura (ROM), Memória de Acesso Aleatório (RAM), Timers e Contadores, Portas de E/S (Entrada/Saída), Interface de Comunicação Serial, Circuito Oscilador e um Mecanismo de Interrupção, como exposto em Figura 3, portanto reunindo em um único chip todos os componentes necessários para leitura e processamento de dados, proporcionando a capacidade de controlar, monitorar e interagir com diversos dispositivos e sistemas de forma eficiente e compacta.

Figura 3. Estrutura básica de um microcontrolador.



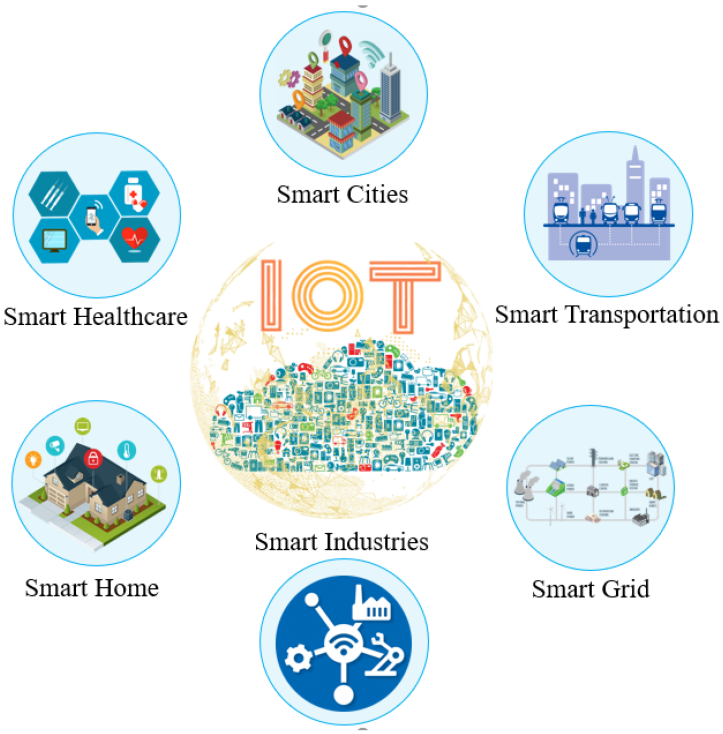
Fonte: (OGUNSEYE et al., 2021)

3.2 Internet das Coisas

A Internet das Coisas é um conceito tecnológico que pressupõe a conexão de objetos físicos à internet, possibilitando a coleta e compartilhamento autônomo de dados entre esses dispositivos, agregando a estes objetos comuns novas habilidades que irão automatizar tarefas, coleta de dados, criação de rotinas precisas e com direcionamento otimizado,gerando novas oportunidades (SANTOS, 2021).

A essência da IoT reside nos dispositivos conectados, que podem incluir sensores de temperatura, câmeras, medidores inteligentes, entre outros. Esses dispositivos se comunicam principalmente por meio de tecnologias sem fio, possibilitando a troca direta de informações entre eles e a transmissão de dados para sistemas de processamento. De acordo com (ZIKRIA, 2021) as aplicações da IoT são diversas e incorporam multimídia em tempo real, sistemas de saúde baseados em IoT, indústrias inteligentes de próxima geração baseadas em IoT e agricultura inteligente de próxima geração baseada em IoT, como mostrado na Figura 4.

Figura 4. Diferentes aplicações da IoT



Fonte: (ZIKRIA, 2021)

Outro aspecto importante para a IoT, além de sua aplicabilidade são as plataformas IoT, que segundo a definição de (FAHMIDEH; ZOWGHI, 2020) são um conjunto de entidades habilitadas por tecnologia, incluindo objetos físicos inteligentes assim como

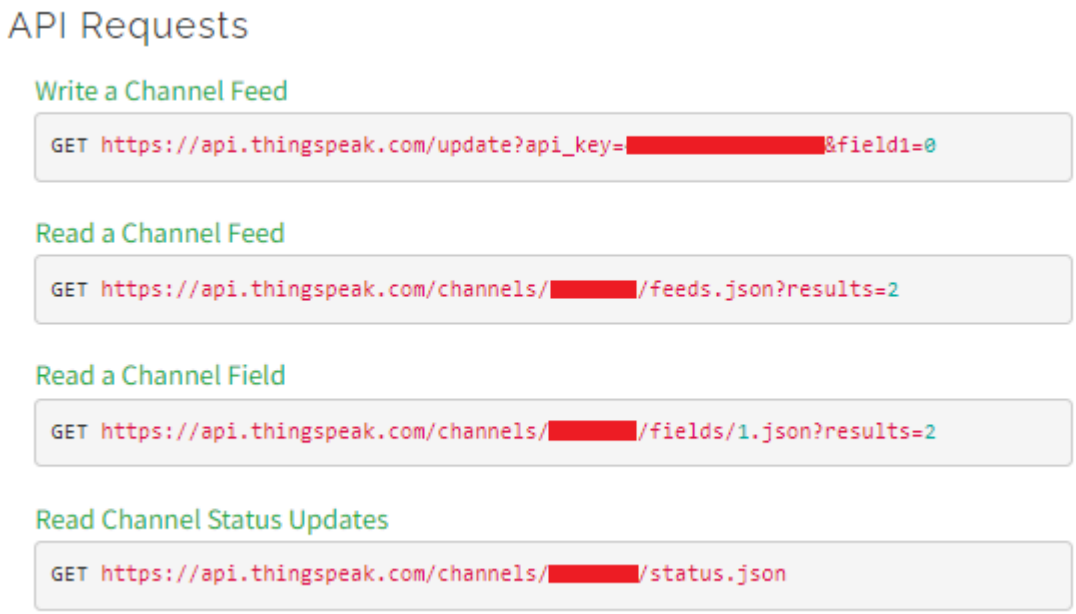
serviços e sistemas de software que estão conectados e trabalham juntos. Desse modo, as plataformas IoT desempenham um papel crucial, pois são responsáveis por receber, processar e armazenar os dados gerados pelos dispositivos conectados, e podem ser baseadas em tecnologias em nuvem, servidores locais ou uma combinação de ambas.

3.2.1 ThingSpeak

Um exemplo de plataforma IoT é o *ThingSpeak*, que segundo (THE MATHWORKS, INC, 2024) é um serviço de plataforma de análise de IoT que permite que agregação, visualização e análise de fluxos de dados ao vivo na nuvem. que fornece serviços de armazenamento, análise e visualização de dados provenientes de dispositivos conectados à internet. Desenvolvido pela MathWorks, o *ThingSpeak* oferece uma solução aberta para coletar, analisar e distribuir dados gerados por sensores e outros dispositivos IoT.

A plataforma é baseada em nuvem e permite que os usuários criem canais específicos para seus dispositivos. Cada canal é estruturado como uma stream de dados que pode ser configurado para aceitar leituras de sensores em intervalos regulares. A integração com a *ThingSpeak* é geralmente feita por meio de sua API (Interface de Programação de Aplicações), facilitando a comunicação entre os dispositivos e a plataforma. Ainda de acordo com (THE MATHWORKS, INC, 2024), o *ThingSpeak* utiliza API REST (Representational State Transfer), que é um modelo de solicitação-resposta que se comunica por meio do HTTP, utilizando chamadas como GET e POST para ler e escrever dados de canal, dentre outras operações. As operações mais básicas da API REST do *ThingSpeak* estão presentes no Dashboard do canal criado, como exposto na Figura 5.

Figura 5. Requisições básicas da API REST do ThingSpeak



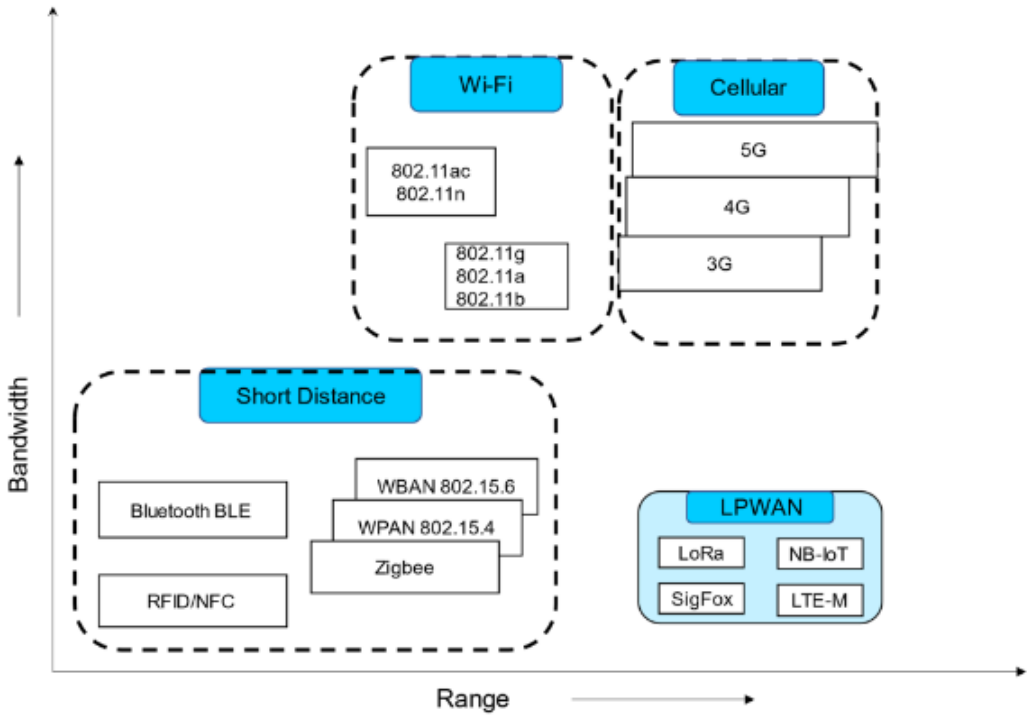
Fonte: Autor

3.3 Redes LPWAN

Redes *Low-Power Wide-Area Network* são um tipo de rede sem-fio projetadas para cobrir uma extensa área a uma baixa taxa de transferência de bits (ROSSATO; SPANHOL; CAMARGO, 2020), conectando dispositivos de baixo consumo energético a longas distâncias. De acordo com (SINGH, 2020) as redes LPWAN têm como características a ampla cobertura, implementação escalável, alta eficiência em termos de energia e são econômicas em relação aos custos de gerenciamento, operação e conjunto de chips de rádio, portanto a característica principal dessas redes é a capacidade de fornecer uma cobertura extensa com um consumo de energia extremamente baixo, em áreas urbanas ou rurais. Essa eficiência energética é crucial para dispositivos alimentados por baterias ou por energia colhida do ambiente, permitindo uma vida útil prolongada para os nós da rede.

As aplicações das redes LPWAN são diversas e incluem monitoramento remoto, rastreamento de ativos, agricultura inteligente, cidades inteligentes, entre outras, visto que segundo (ALBEYBONI; ALI, 2023) essas redes foram criadas especificamente para aplicações que exigem o envio de algumas mensagens curtas todos os dias em longo alcance, como exposto na Figura 6. Sua capacidade de oferecer uma cobertura ampla e eficiência energética torna essas redes fundamentais para a expansão das aplicações da Internet das Coisas em escala global.

Figura 6. Comparação de Cobertura e Largura de banda das tecnologias sem fio.



Fonte: (ALBEYBONI; ALI, 2023)

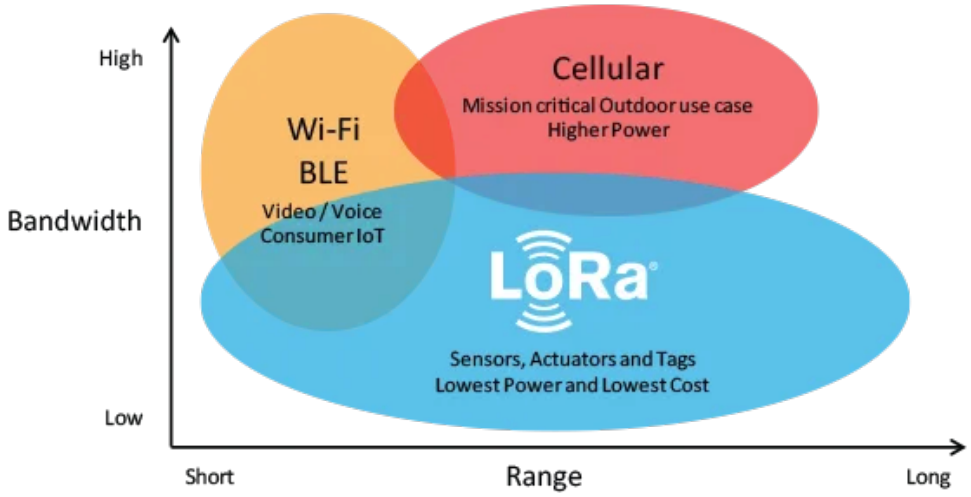
3.3.1 LoRa

LoRa (*Long Range*) refere-se a uma técnica de modulação de rádio que essencialmente consiste em manipular ondas de rádio para codificar informações usando um formato de símbolos múltiplos, através de uma tecnologia de espectro expandido por chirp (chirp spread spectrum) (TREND MICRO INCORPORATED, 2024). Projetada para fornecer uma solução de conectividade eficiente em termos de energia e alcance, a tecnologia LoRa se destaca por permitir comunicação de longo alcance entre dispositivos de baixo consumo energético, com aplicações predominantemente na Internet das Coisas (IoT), sendo uma alternativa notável no campo das Redes LPWAN.

De acordo com (THE THINGS INDUSTRIES, 2024) LoRa é ideal para aplicações que transmitem pequenos pacotes de dados com baixas taxas de bits, e a transmissão pode ocorrer a distâncias maiores em comparação com tecnologias como WiFi, Bluetooth ou ZigBee, tornando-o adequado para sensores e atuadores que operam em modo de baixo consumo de energia. Tendo em vista essa característica do LoRa, suas aplicações tornam propício o oferecimento de uma cobertura de comunicação de longo alcance, atingindo distâncias consideráveis, muitas vezes superiores a vários quilômetros em condições ideais, sendo até cinco quilômetros em áreas urbanas e até 15 quilômetros ou mais em áreas rurais, em linha de visão (SEMTECH, 2024), o que é um aspecto importante para aplicações em

áreas urbanas, rurais ou industriais, onde a conectividade em larga escala é essencial. Um comparativo do alcance e da largura de banda do LoRa pode ser verificado em Figura 7.

Figura 7. Comparação de Cobertura e Largura de banda da tecnologia LoRa com outras comunicações sem fio.



Fonte: (THE THINGS INDUSTRIES, 2024)

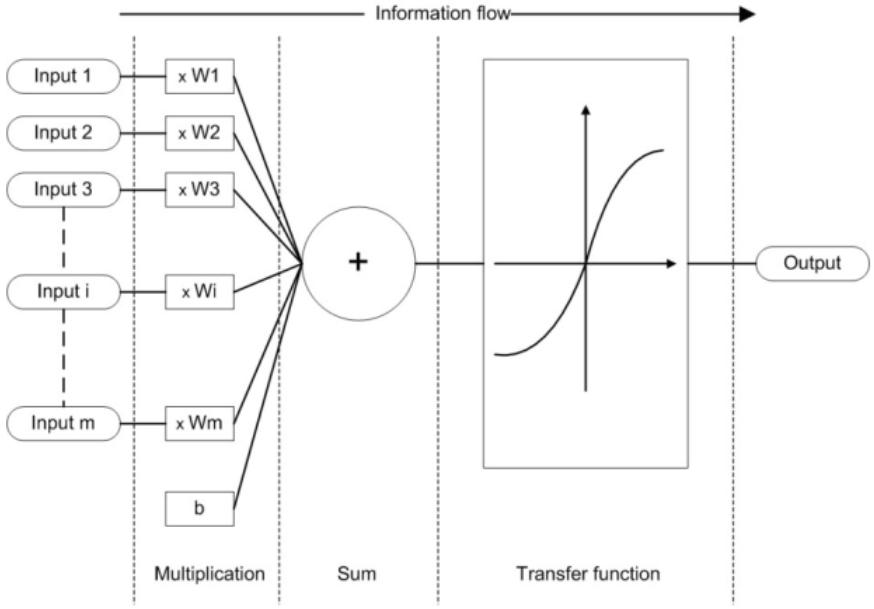
O LoRa opera na faixa de frequência ISM (*Industrial, Scientific and Medical*), podendo ser operado nas faixas de subgigahertz isentas de licença, como, por exemplo, 915 MHz, 868 MHz e 433 MHz, e podem também ser operado em 2,4 GHz para alcançar taxas de dados mais altas em comparação com as faixas de subgigahertz, à custa do alcance (THE THINGS INDUSTRIES, 2024). Além disso, o LoRa utiliza a tecnologia CSS (*Chirp Spread Spectrum*), que possui como características performance de longo alcance com alta confiabilidade, resistência forte a interferências de rádio e baixo consumo de energia (INPIXON, 2024), o que contribui para a resistência a interferências e melhora a capacidade de penetração em obstáculos físicos, como edifícios urbanos.

Os dispositivos LoRa são comumente utilizados em conjunto com o LoRaWAN, que é um protocolo de rede de Área Ampla e Baixo Consumo de Energia (LPWA), projetado para conectar sem fio dispositivos alimentados por bateria à internet em redes regionais, nacionais ou globais (LORA ALLIANCE, 2024), sendo esse protocolo de grande importância para a implementação da tecnologia LoRa no IoT por permitir a padronização do uso da mesma, porém é possível fazer uso do LoRa em comunicações diretas ponto a ponto sem o uso do protocolo LoRaWAN.

3.4 Redes Neurais Artificiais

Redes Neurais Artificiais (RNAs) são um modelo matemático que busca simular a estrutura e funcionalidades de redes neurais biológicas, tendo como bloco básico de construção o neurônio artificial, ou seja, um modelo matemático simples com capacidade de multiplicação, soma e ativação (KRENKER; BEŠTER; KOS, 2011), como observado na Figura 8. Esses blocos básicos colaboram na realização de operações complexas, mediante a ponderação e combinação de entradas, fornecendo um meio para lidar com problemas complexos orientados a padrões, tanto do tipo categorização quanto de séries temporais, na forma de análise de tendências (WALCZAK, 2019).

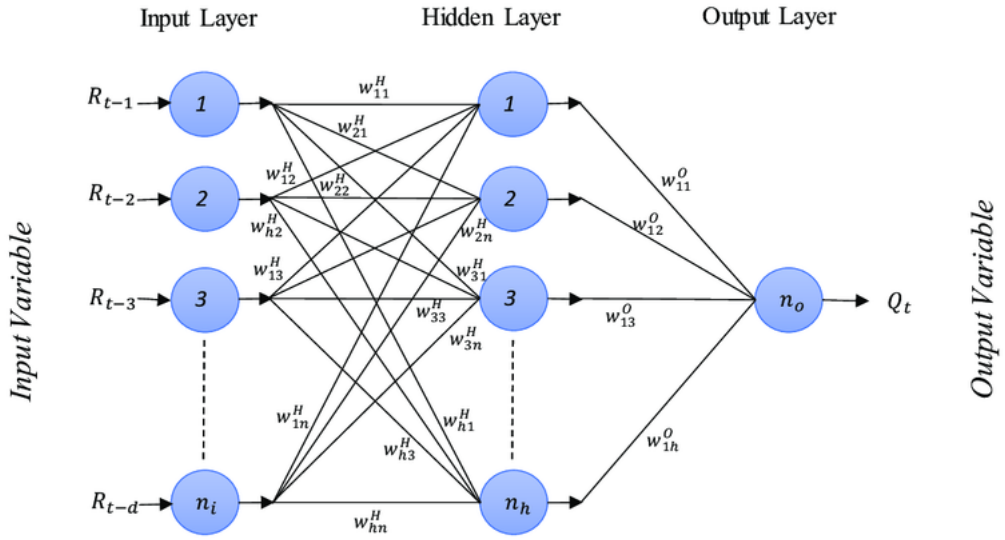
Figura 8. Princípio de funcionamento do neurônio artificial.



Fonte: (KRENKER; BEŠTER; KOS, 2011)

Existem várias implementações baseadas no conceito de RNA, dentre elas destaca-se o MLP (Perceptron de Múltiplas Camadas), que é o tipo mais conhecido e mais frequentemente utilizado de rede neural, onde na maioria das ocasiões os sinais são transmitidos dentro da rede em uma direção, da entrada para a saída (POPESCU, 2009), conhecidas também como Redes *Feedforward* (FFW). O perceptron possui a capacidade de receber múltiplas entradas ponderadas, realizar a soma dessas entradas ponderadas a um termo de bias e aplicar uma função de ativação para produzir a saída. Essa arquitetura elementar é organizada em camadas, sendo uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída, como representado em Figura 9.

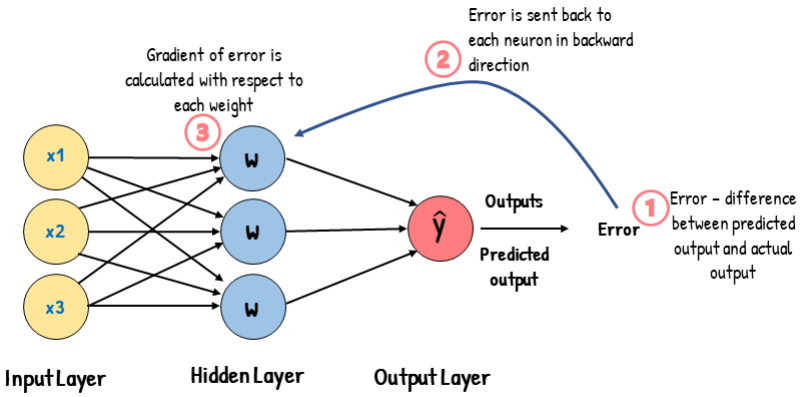
Figura 9. Estrutura geral de um MLP FFW.



Fonte: (KHAN et al., 2021)

Um dos algoritmos mais utilizados para treinamento das RNAs é o *Backpropagation*, que segundo (CARVALHO, 2009) opera em uma sequência de dois passos, onde primeiro um padrão é apresentado à rede para obter uma resposta na saída, que é utilizada para calcular o erro em relação ao valor correto conhecido. Em seguida, o erro é retropropagado pelas camadas, atualizando os pesos das camadas intermediárias da RNA, como mostra a Figura 10.

Figura 10. Exemplo de Backpropagation.



Fonte: (KALIRANE, 2023)

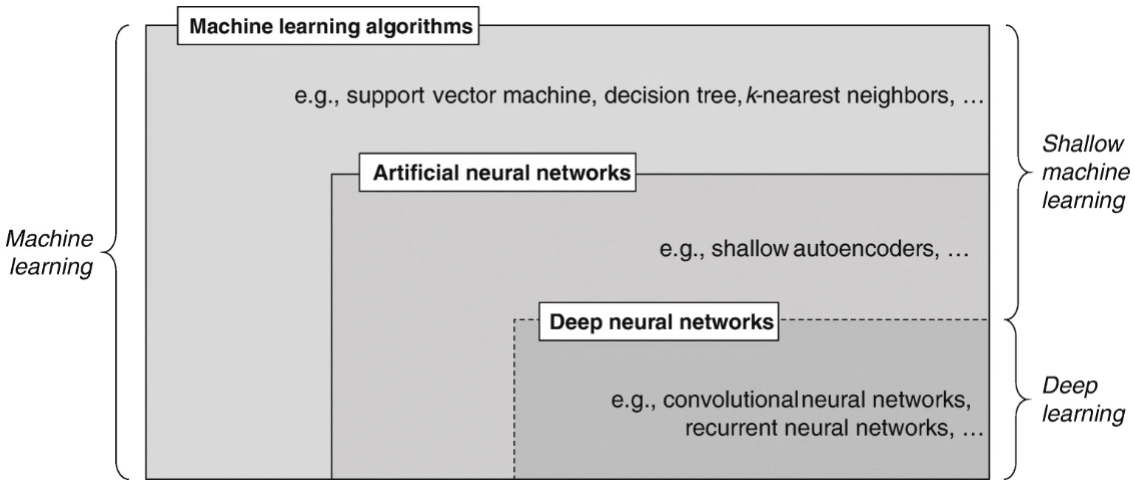
Diversas aplicações de RNA surgiram codificadas em diversas linguagens de progra-

mação, dentre elas destaca-se o *Python*, que é uma linguagem de programação interpretada, orientada a objetos e de alto nível, com semântica dinâmica (PSF, 2024). Essa linguagem de programação passou a se destacar no campo de machine learning devido à sua extensa gama de bibliotecas e frameworks especializados, proporcionando facilidade de uso para o rápido desenvolvimento de algoritmos e modelos, enquanto a vasta quantidade de recursos disponíveis facilita a implementação eficiente de tarefas complexas de aprendizado de máquina. Outra facilidade referente ao Python é a presença de ambientes virtuais onlines que permitem criação e execução de scripts utilizando recursos computacionais de servidores na nuvem como o *Kaggle*, que é um ambiente que possibilita que cientistas de dados e outros desenvolvedores participem de competições de aprendizado de máquina, escrevam e compartilhem código, e hospedem conjuntos de dados (USMANI, 2017).

3.5 Deep Learning

Deep Learning, ou Aprendizado Profundo, é um ramo da área de aprendizado de máquina e inteligência artificial (SARKER, 2021) que se fundamenta na utilização de Redes Neurais Artificiais profundas para realizar tarefas complexas de aprendizado e representação de dados. De acordo com (JANIESCH; ZSCHECH; HEINRICH, 2021) as redes neurais profundas, geralmente consistem em mais de uma camada oculta, organizadas em arquiteturas de rede profundamente aninhadas, e geralmente contêm neurônios avançados em contraste com redes neurais artificiais simples. A relação entre os conceitos de aprendizado de máquina, rede neural artificial e Redes Neurais Profundas pode ser observada na Figura 11.

Figura 11. Diagrama de Venn dos conceitos e classes de aprendizado de máquina.



Fonte: (JANIESCH; ZSCHECH; HEINRICH, 2021)

A característica distintiva do *Deep Learning* é a presença de redes neurais profundas, geralmente compostas por várias camadas ocultas, o que proporciona à arquitetura a capacidade de aprender representações de dados em diferentes níveis de abstração. Devido à essa capacidade, o *Deep Learning* tem demonstrado notável sucesso em diversas áreas, e pode ser aplicado em diversas áreas, como reconhecimento de imagem para identificar objetos e características, processamento de linguagem natural para compreender o significado de texto em chatbots, previsões de tendências e conversão de texto para imagem (GOOGLE, 2024).

3.5.1 Redes Neurais Convolucionais

Redes Neurais Convolucionais (CNNs) representam uma classe especializada de redes neurais profundas, e conforme (DATA SCIENCE ACADEMY, 2022) são um algoritmo de Aprendizado Profundo que pode captar uma imagem de entrada, atribuir importância, tais como pesos e vieses que podem ser aprendidos, a vários aspectos e objetos da imagem. Essa arquitetura é inspirada pela sensibilidade local e orientação seletiva do cérebro, resultando em uma rede neural que extrai características relevantes da entrada implicitamente (SOARES, 2018).

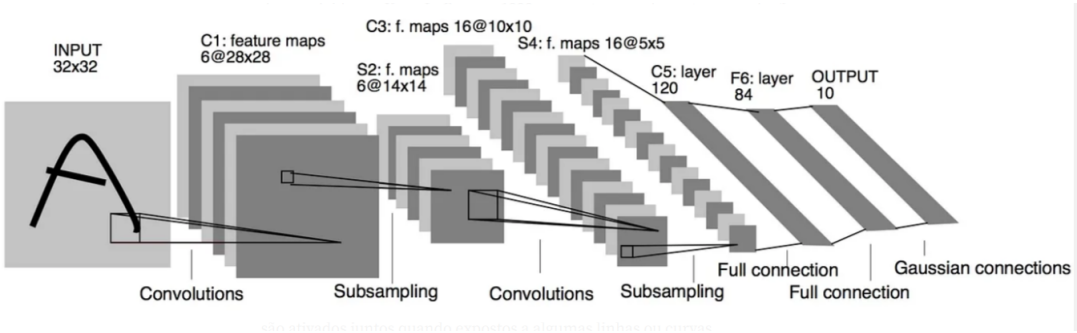
De acordo com (IBM, 2024), a camada convolucional é o bloco de construção central de uma CNN e é onde ocorre a maioria dos cálculos, e exige alguns componentes, que são dados de entrada, um filtro e um mapa de feições. As camadas convolucionais realizam operações de convolução para extrair características locais, e os filtros são treinados para reconhecer padrões específicos, como bordas, texturas ou formas, possibilitando a aprendizagem de características significativas da imagem.

As CNNs possuem outras camadas importantes, como a cadama de *pooling* ou *subsampling*, frequentemente aplicada após as camadas convolucionais, que reduz a dimensionalidade do mapa de características, preservando as informações mais relevantes. As CNNs incluem também camadas totalmente conectadas, que consolidam as informações aprendidas nas camadas convolucionais para a tomada de decisões finais, e funções de ativação que introduzem não-linearidades ao modelo. A capacidade de compartilhamento de parâmetros e a utilização de pesos compartilhados nas camadas convolucionais tornam as CNNs eficientes e capazes de lidar com a complexidade de dados visuais e atributos posicionais.

Na Figura 12 é possível observar um exemplo comum de implementação de CNN, na qual a primeira camada é a de entrada, onde uma a imagem é inicialmente inserida na rede. Em seguida, temos as camadas de convolução, que aplicam operações de convolução para extrair características da imagem. Essas características são então passadas para as camadas de subamostragem, ou *pooling*, que reduzem a dimensionalidade dos dados, enquanto retêm as informações mais importantes, sendo que o processo de convolução

e subamostragem pode ocorrer várias vezes, cada vez extraindo características mais complexas. Finalmente, as características extraídas são integradas em camadas totalmente conectadas para reconhecimento de padrões, onde a última camada é a camada de saída, que produz as categorias classificadas.

Figura 12. Camadas comumente presentes em uma CNN.



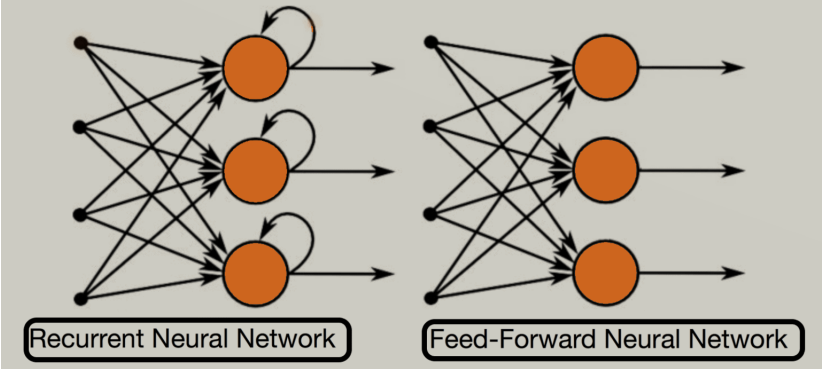
Fonte: (ALVES, 2018)

3.5.2 Redes Neurais Recorrentes

Redes Neurais Recorrentes (RNNs) são modelos de aprendizado profundo treinados para processar e converter uma entrada de dados sequencial em uma saída de dados sequencial específica (AMAZON, 2023), demonstrando notável aptidão para a modelagem e análise de sequências temporais e dados sequenciais. Em contraste com redes neurais tradicionais, as RNNs incorporam mecanismos de realimentação, guardando um histórico das entradas anteriores para cada saída (SILVA, 2022), permitindo a consideração de informações contextuais anteriores durante o processamento de entradas subsequentes.

Esse design habilita a captura de dependências temporais de longo alcance, sendo particularmente eficaz em tarefas que envolvem sequencialidade, como reconhecimento de fala, tradução automática, previsão de séries temporais e processamento de linguagem natural. Diferente das redes *Feedforward*, a arquitetura das RNNs consiste em unidades recorrentes que mantêm estados internos, sendo capazes de memorizar informações passadas e incorporá-las no processamento futuro, como visto na Figura 13. Apesar de suas vantagens, RNNs enfrentam desafios, como a propensão ao desvanecimento ou explosão do gradiente, o que pode comprometer o aprendizado de dependências a longo prazo.

Figura 13. Comparação da estrutura de uma rede FFW com uma RNN.

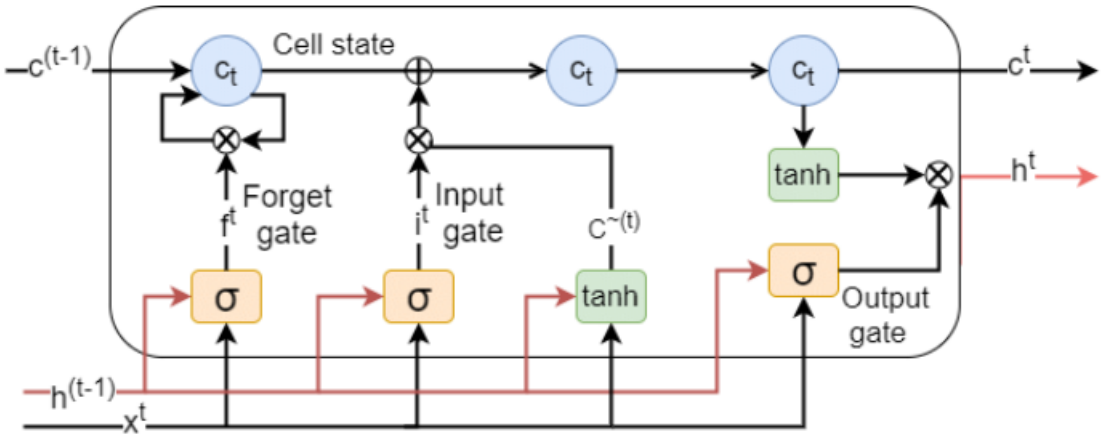


Fonte: (TONDAK, 2023)

A rede Long Short-Term Memory networks (LSTM) é uma variante especial das RNNs capaz de aprender dependências de longo prazo, projetada especificamente para evitar o problema de explosão do gradiente (VASCO, 2020), portando unidades de células de memória, portões de entrada, esquecimento e saída. Cada unidade de célula de LSTM é projetada para armazenar e recuperar informações com base em comandos de entrada e decisões do passado, possuindo estruturas denominadas gates ou portões, para adquirir manter ou esquecer determinada porção da informação retida, mitigando assim o problema do desvanecimento do gradiente. Essa arquitetura permite que as LSTMs capturem dependências temporais complexas e aprendam padrões de longo prazo de maneira mais eficaz do que as RNNs tradicionais, tornando-as particularmente adequadas para aplicações que exigem a modelagem de sequências temporais extensas.

A Figura 14 representa a estrutura interna de uma célula LSTM, na qual estão presentes suas estruturas específicas, como seus portões e estados que permitem o fluxo e a transformação de dados. Nesta, o Portão de Esquecimento está conectado ao estado da célula anterior e à entrada atual, e o Portão de Entrada recebe as informações para o presente estado. Os estados da célula transformam-se, do estado anterior, para o atual através da modulação do portão de esquecimento e da adição de novas informações da entrada modulada pela função de ativação, e o Portão de Saída determina qual parte do estado atual é emitida, notando-se a presença de duas saídas da célula: uma representando o estado da mesma definido pelos Portões de Esquecimento e Entrada e outra que corresponde à um estado Oculto da célula, que é utilizado de forma recursiva nos outros Portões para o cálculo do próximo estado.

Figura 14. Estrutura de uma célula LSTM.



Fonte: (JENKINS et al., 2018)

3.6 NSGA-II

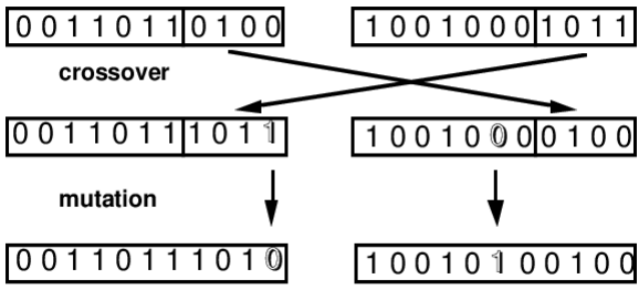
O *Non-dominated Sorting Genetic Algorithm II* (NSGA-II) representa uma técnica avançada de otimização multiobjetivo baseada em algoritmos genéticos, desenvolvida para superar as limitações de abordagens anteriores, como o NSGA original, tais como a complexidade computacional do problema e a ausência de técnicas de seleção diversificada e elitismo (DEB, 2002), permitindo que o novo algoritmo destaque-se ao lidar eficientemente com problemas de otimização que envolvem múltiplos objetivos conflitantes.

A principal característica do NSGA-II é a implementação do conceito de classificação não-dominada, na qual uma solução viável é dita não-dominada ou eficiente, se não for dominada por nenhuma outra solução viável no espaço de busca (PÉREZ, 2012), onde uma solução dominada é definida como inferior ou estritamente pior em relação a outra em pelo menos um critério, sem ser superior em nenhum critério. Todas as soluções ditas não-dominadas pertencentes a um conjunto denominado fronteira de Pareto (SIQUEIRA, 2019), que se trata de uma população de soluções eficientemente distribuída, oferecendo ao tomador de decisões um conjunto de opções não-dominadas.

O processo evolutivo do NSGA-II envolve a criação de descendentes através de operadores genéticos tais como crossover e mutação, presentes na Figura 15, seguido pela avaliação das soluções candidatas em relação aos objetivos do problema, de forma iterativa dado um número definido de gerações. As soluções são então classificadas em fronteiras de Pareto, e a população é atualizada a cada geração utilizando um processo de elitismo para preservar as soluções de alta qualidade e aplicando a mutação probabilística para a manutenção da diversidade dos indivíduos gerados. Uma representação desses operadores

em formato de Pseudocódigo pode ser observada na Figura 16.

Figura 15. Exemplo de Crossover e Mutação de dois indivíduos em um algoritmo genético.



Fonte: (PADRó; OZóN; APLICADA, 1995)

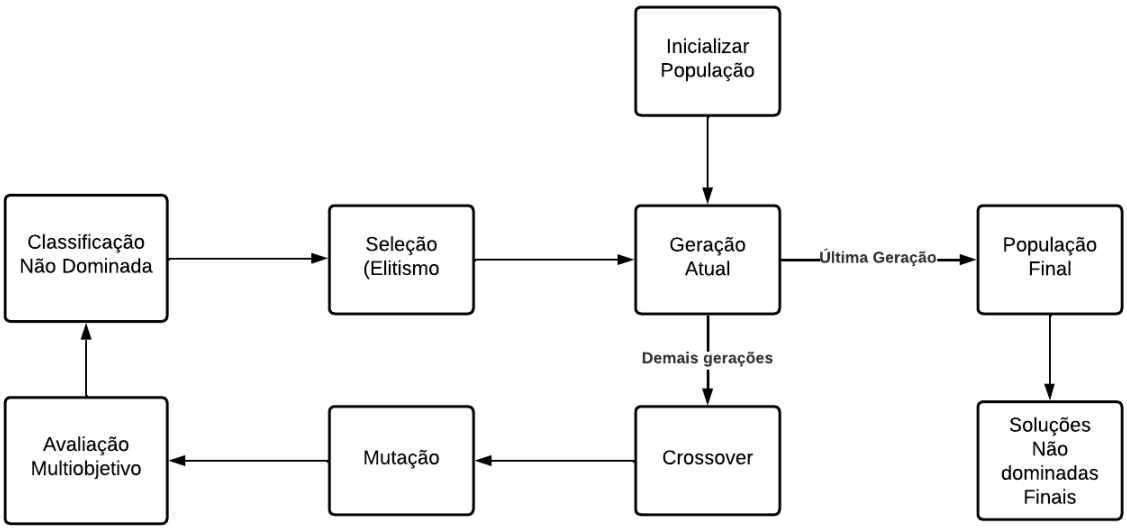
Figura 16. Pseudocódigo de Crossover e Mutação de indivíduos representados por vetores.

```
//Crossover
Função Crossover(Pai1, Pai2):
    // Pai1 e Pai2 são cromossomos parentais
    Filho1 = NovoCromossomo()
    Filho2 = NovoCromossomo()

    // Escolher um ponto de crossover aleatório
    PontoCrossover = Aleatorio(1, TamanhoCromossomo - 1)

    // Realizar crossover para criar os filhos
    Para i de 0 até PontoCrossover:
        Filho1[i] = Pai1[i]
        Filho2[i] = Pai2[i]

    Para i de PontoCrossover + 1 até TamanhoCromossomo - 1:
        Filho1[i] = Pai2[i]
        Filho2[i] = Pai1[i]

    Retornar Filho1, Filho2
Fim Função

//Mutação
Função Mutacao(Cromossomo, TaxaMutacao):
    // Cromossomo é o cromossomo a ser mutado
    // TaxaMutacao é a probabilidade de mutação para cada gene

    Para cada gene no Cromossomo:
        // Gerar um número aleatório entre 0 e 1
        Aleatorio = GerarNumeroAleatorio()

        // Se o número aleatório for menor que a taxa de mutação, mutar o gene
        Se Aleatorio < TaxaMutacao:
            // Realizar a mutação (por exemplo, alterar o valor do gene)
            Cromossomo[gene] = NovoValorAleatorio()

    Retornar Cromossomo
Fim Função
```

Fonte: Autor

Um exemplo de implementação desta técnica de otimização pode ser visualizado no diagrama da Figura 17, onde o processo se inicia com a criação de uma população inicial de soluções potenciais, e em cada iteração as soluções são classificadas com base em seu desempenho em múltiplos objetivos, e os indivíduos de melhor desempenho são selecionados para formar um pool de acasalamento. Estes indivíduos passam pelas operações genéticas de mutação e crossover para gerar novas soluções potenciais. A adequação dessas novas soluções é avaliada e o processo é repetido até que a população evolua para a geração final. As soluções finais não dominadas são consideradas trocas ótimas entre objetivos conflitantes, compondo assim a fronteira de Pareto das soluções finais.

Figura 17. Diagrama de Blocos da implementação do Algoritmo NSGA II.



Fonte: Autor

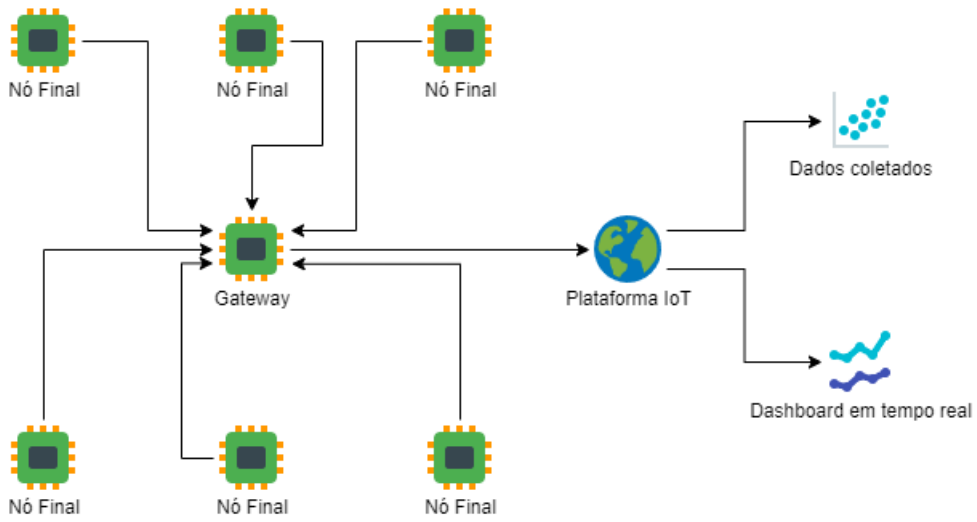
4 Metodologia

4.1 Cenário Proposto

O cenário proposto foi idealizado como uma Rede de Sensores IoT com nós finais distribuídos em um ambiente, coletando informações da temperatura do ambiente, utilizando dois tipos de sensores de temperatura com diferentes consumo de energia. Esses nós finais possuem uma bateria recarregável para manter seu funcionamento, e essa bateria pode ser alimentada por fontes de energia colhidas do ambiente, como solar e eólica de baixa tensão, e desse modo eles também possuem sensores com capacidade de aferir tensão e corrente de uma fonte de corrente contínua. Todas as informações coletadas dos sensores dos nós finais são repassadas à um Gateway que por sua vez repassa esses dados para um plataforma IoT na Internet, tornando os dados acessíveis remotamente para monitoramento.

Como a vida útil do dispositivo está diretamente relacionada ao tempo de descarga da bateria, todos os nós finais possuem a capacidade de medir a tensão da bateria presente, tornando possível verificar o processo de descarga. Os nós finais não necessariamente estarão com todos os sensores mencionados presentes, utilizarão apenas os sensores necessários para o ambiente o qual se encontram, portanto diferentes nós finais irão gerar diferentes curvas de descarga devido à diferentes combinações de sensores conectados. A idealização do cenário geral pode ser visualizado na Figura 18.

Figura 18. Cenário geral da Rede de Sensores IoT

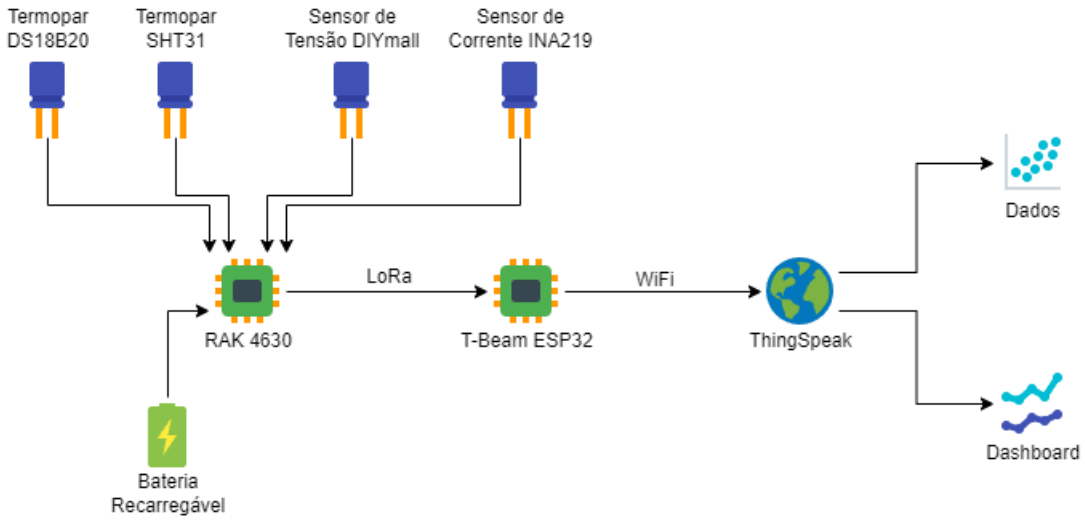


Fonte: Autor

4.2 Prototipagem

Para realizar uma implementação que permita avaliar o consumo de energia dos nós finais desse cenário, foi projetado um nó final e um *gateway*, criando uma rede LPWAN local entre esses componentes e conectando essa rede de sensores à Internet através por meio de conexão WiFi, conforme explicitado na Figura 19, e a lista completa de dispositivos utilizados na prototipagem está presente na Tabela 2.

Figura 19. Diagrama da Prototipagem realizada



Fonte: Autor

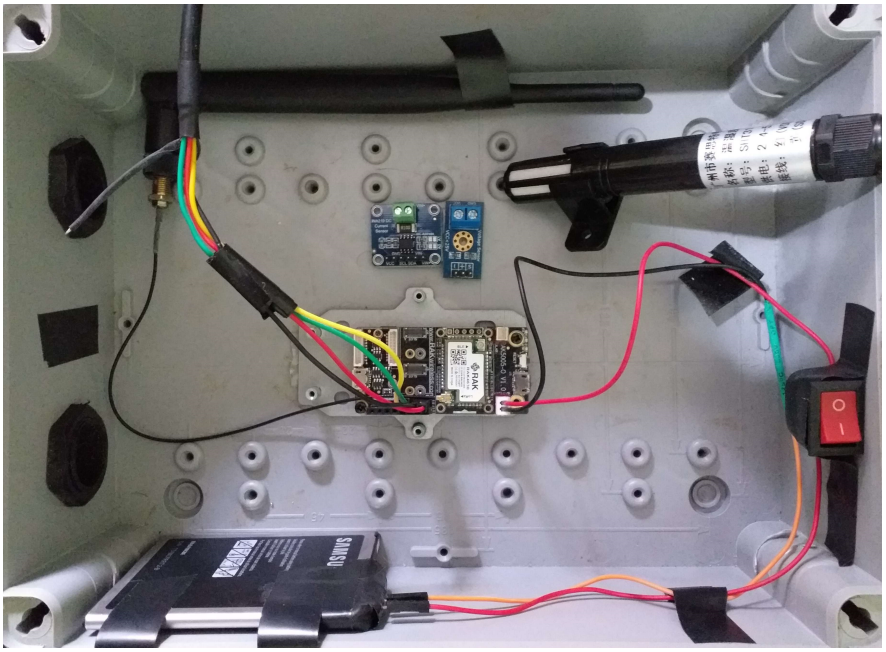
Tabela 2. Relação dos componentes utilizados na implementação.

Nó Final	
RAK4630	Microcontrolador com capacidade de comunicação LoRa
RAK5005-O	Placa de interface para o microcontrolador RAK4630
DS18B20	Sensor de temperatura
SHT31	Sensor de temperatura
Voltímetro DIYmall	Sensor de tensão
INA219	Sensor de corrente
EB-BJ700CBE	Bateria recarregável
Gateway	
TTGO T-Beam ESP32	Microcontrolador com com LoRa e WiFi
Plataforma IoT	
ThingSpeak	Plataforma para armazenamento dos dados

Foi produzido um Nó final com capacidade de manter diferentes sensores conectados ao mesmo, gerando dados que são subsequentemente transmitidos por meio da tecnologia

de comunicação LoRa. Foi utilizado o microcontrolador RAK4630, que é um módulo transceptor de longo alcance de baixo consumo baseado no MCU Nordic nRF52840 que suporta o transceptor LoRa SX1262 da Semtech (RAKWIRELESS TECHNOLOGY LIMITED, 2024), sua implementação pode ser verificada na Figura 20, e os parâmetros utilizados para a configuração da transmissão estão presentes na Tabela 3.

Figura 20. Vista interna do Nó final.



Fonte: Autor

Tabela 3. Parâmetros da trasmissão LoRa utilizados na configuração do Nó final.

Parâmetro	Valor adotado
Frequência	915 MHz
SF	12
Potência de Transmissão	22 dBm
Tamanho do Pacote	59 kB
Frequência de Transmissão	60 s

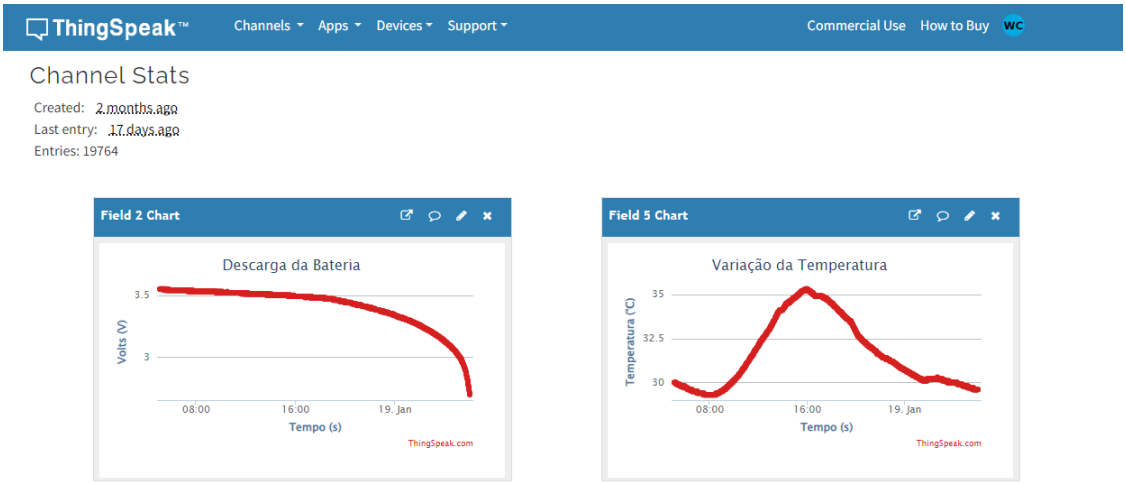
O gateway por sua vez consiste em um dispositivo baseado no TTGO T-Beam, que é uma placa de desenvolvimento ESP32 que utiliza diretamente o chip ESP32 e que integra um SemTech SX1276 ou SX1278 para comunicação LoRa (YEFREMOV, 2024), sendo capaz de receber mensagens via LoRa e encaminhar esses dados para uma plataforma online via WiFi. O *Gateway* utilizado está representado na Figura 21.

Figura 21. Microcontrolador T-Beam utilizado como Gateway.

Fonte: Autor

A topologia adotada na configuração da rede é caracterizada por uma conexão ponto a ponto LoRa entre o nó final e o *Gateway*, no qual um pacote de dados é transmitido a cada intervalo de um minuto para o *Gateway*, que o retransmite para a plataforma ThingSpeak na Internet, através de uma conexão WiFi presente no mesmo ambiente em que o *Gateway* se encontrava, tornando possível o acompanhamento da leitura dos sensores presentes no Nó final no Dashboard da plataforma, como mostra a Figura 22.

Figura 22. Dashboard da plataforma ThingSpeak



Fonte: Autor

4.3 Dados coletados

Foram realizados experimentos de descarga de bateria com diferentes sensores conectados ao Nó final, objetivando a visualização de curvas de descarga distintas devido aos padrões de consumo energético específicos gerados pelas combinações diferentes de sensores. Ressalta-se que, apesar dos sensores escolhidos para a prototipagem estarem coesos à aplicação definida no cenário geral proposto, o único objetivo dos testes é gerar diferentes curvas de descarga para o treinamento do modelo estimador de descarga, portanto os dados coletados pelos sensores acoplados ao Nó final não se referem à nenhum evento relevante para o presente estudo e nem agregam nenhum dado importante à análise além do fato dos mesmos estarem corretamente conectados ao dispositivo e operando de acordo com suas respectivas referências técnicas, contribuindo para o consumo de energia. Foram realizados cinco testes de descarga combinando os sensores mencionados na Tabela 2, a relação dos testes realizados pode ser verificada na Tabela 4.

Tabela 4. Lista de experimentos de Descarga realizados.

Experimento	Sensores presentes no Nó final
Descarga A	DS18B20
Descarga B	DS18B20 e Voltímetro DIYmall
Descarga C	SHT31
Descarga D	Voltímetro DIYmall
Descarga E	SHT31 e INA219

Os dados referentes à tensão da bateria recarregável e as leituras dos sensores foram armazenadas na plataforma IoT, da qual puderam ser extraídos em um formato tabular, possuindo identificadores de timestamp e identificadores de pacote referente ao experimento realizado, tornando fácil o acesso e interpretação dos dados pela linguagem de programação utilizada, permitindo o tratamento e classificação das informações de cada experimento. Um trecho dos dados coletados e o cabeçalho da tabela gerada pelo *ThingSpeak* pode ser visualizado na tabela Tabela 5

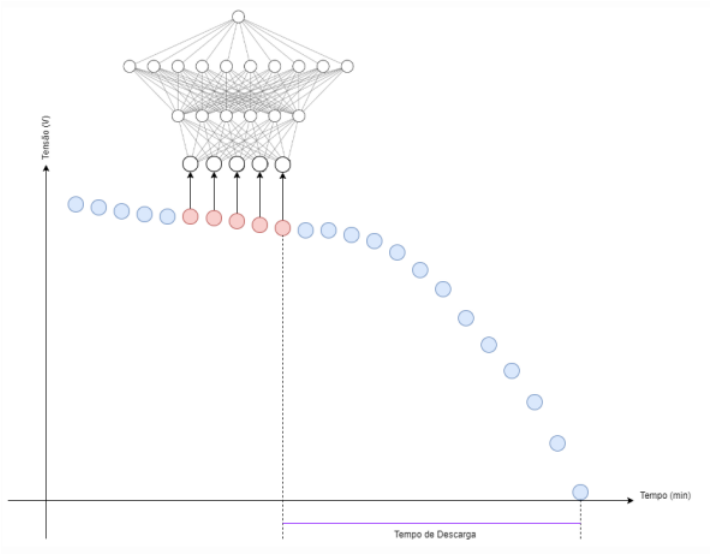
Tabela 5. Exemplo de dados tabulares extraídos da API ThingSpeak

created_at	entry_id	field1	field2	field3
2023-12-15T12:57:07-03:00	1	0	4.10	0.01
2023-12-15T12:57:49-03:00	2	0	4.10	0.01
2023-12-15T12:58:51-03:00	3	0	4.09	0.01
2023-12-15T12:59:54-03:00	4	0	4.10	0.00
2023-12-15T13:00:57-03:00	5	0	4.10	0.00
2023-12-15T13:01:59-03:00	6	0	4.10	0.01
2023-12-15T13:03:01-03:00	7	0	4.10	0.01

4.4 Treinamento do Modelo Estimador de Descarga

Foi desenvolvido um script em Python, no ambiente virtual Kaggle, presente em toda sua extensão no Apêndice A, com o objetivo de captar os dados tabulares provenientes da ThingSpeak e utilizá-los no desenvolvimento de um modelo de Rede Neural Artificial com elementos Deep Learning para estimação do tempo de descarga, a partir da observação de um segmento da curva. O princípio desse método trata-se de extrair um número de pontos consecutivos da curva de descarga, que será denominado janela de observação, observar os atributos posicionais dessa fração da curva e estimar o tempo de descarga a partir do último ponto da janela de observação, como demonstrado na Figura 23.

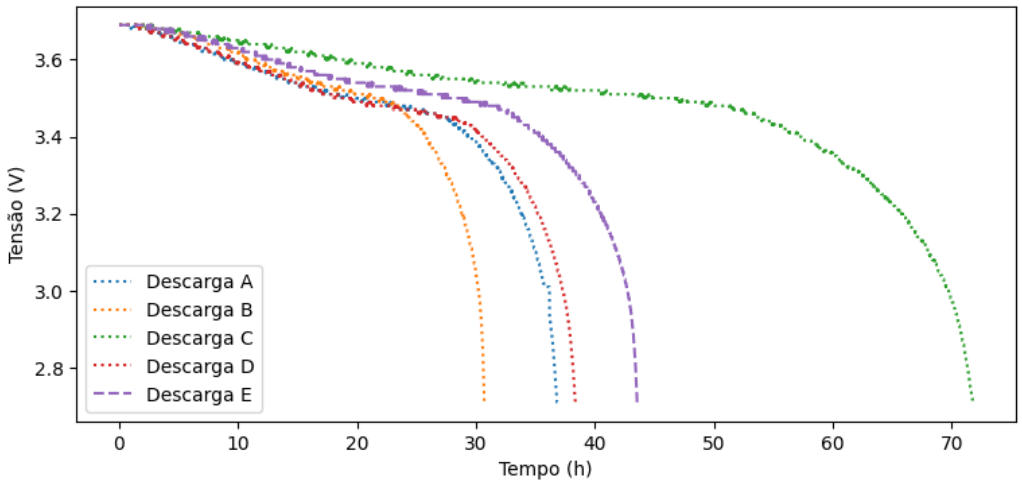
Figura 23. Janela de observação representada pelo segmento repassado como entrada do modelo, e o Tempo de Descarga representada pela distância temporal da janela ao fim da curva de descarga.



Fonte: Autor

Dessa forma, a primeira etapa do processo é o carregamento dos conjuntos de dados associados a experimentos, e em seguida, realiza-se um processo de pré-processamento, filtrando e separando os dados em curvas específicas de descarga relacionadas a diferentes experimentos. Além disso, os timestamps presentes em cada pacote são convertidos para segundos, gerando uma escala temporal associada a cada valor de tensão assumido pela bateria durante a descarga. As curvas geradas podem ser observadas na Figura 24.

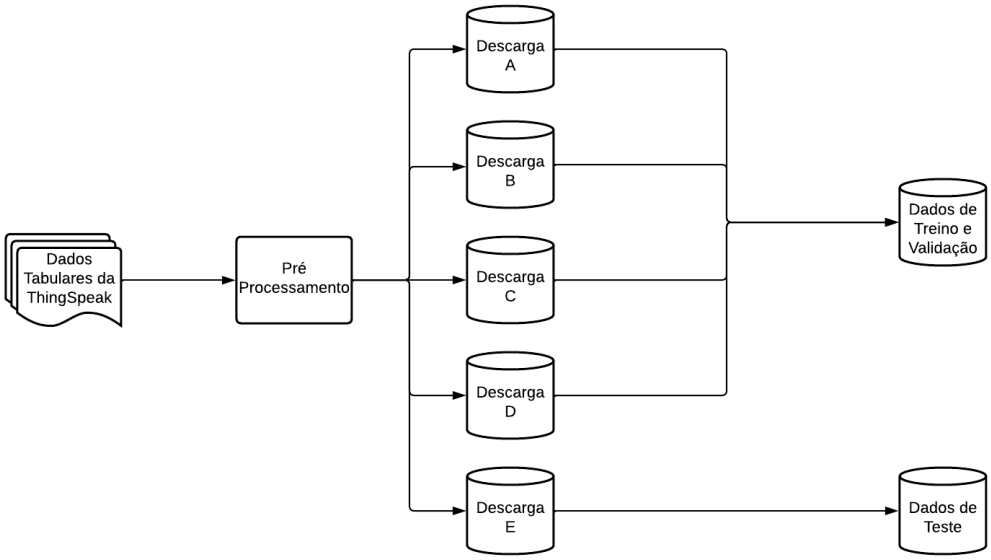
Figura 24. Curvas de descarga geradas pelos experimentos



Fonte: Autor

Com o objetivo de permitir com que o algoritmo NSGA-II avalie diferentes tamanhos de janela de observação da curva, uma série de funções auxiliares são definidas para gerar conjuntos de dados de treino e teste baseados em um dado números de pontos, que correspondem ao tamanho do segmento observado da curva descarga, portanto, correspondendo também ao número de neurônios da camada de entrada do modelo. Como representado na Figura 25, utilizou-se quatro das cinco curvas obtidas para realização do treino e validação dos modelos gerados, e reservou-se a última curva apenas para base de dados de teste para que a validação dos modelos ocorressem em instâncias que nunca foram utilizadas durante o processo de treinamento.

Figura 25. Geração dos conjuntos de Treino e Teste para os Modelos



Fonte: Autor

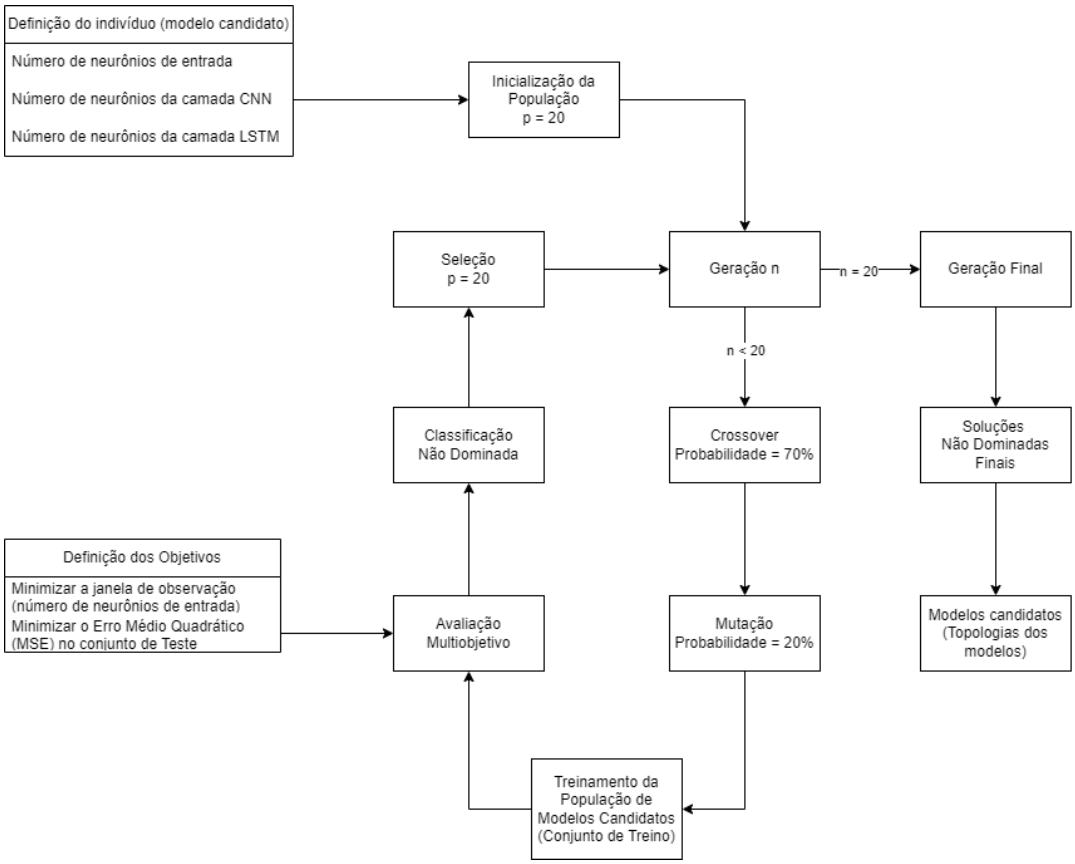
Objetivando o desenvolvimento de um modelo robusto e adaptável, definiu-se que o modelo estimador irá possuir uma camada escondida de CNN, pois a capacidade dessa rede de reconhecer atributos posicionais tornaria propício o reconhecimento de variações ou tendências na janela analisada que auxiliem o modelo a estimar o tempo de descarga. Foi definido que o modelo também possuirá uma segunda camada escondida do tipo LSTM, pois esse tipo de rede recorrente possui a capacidade de armazenar informações importantes de sequências, como sugere o termo memória de longo prazo. Definido o arcabouço do modelo, percebe-se que ele irá possuir uma camada de entrada, correspondente ao tamanho da janela de observação, um camada escondida CNN, uma camada escondida LSTM e um neurônio na camada de saída, correspondente ao tempo estimado pelo modelo.

Para aplicar a otimização multiobjetivo com o intuito de definir a quantidade de neurônios na camada de entrada e nas duas camadas escondidas, de modo a obter a estimativa mais precisa, definiu-se que um indivíduo é equivalente à um modelo com as três camadas citadas com número de neurônios variáveis, e que os objetivos da otimização seriam minimizar o tamanho da janela de observação e a diferença do resultado estimado dos modelos em relação ao resultado real do conjunto de teste, como exposto na Tabela 6. Após a definição do Indivíduo e dos atributos a serem otimizados, utilizou-se o NSGA-II para determinar o melhor indivíduo, inicializando uma população desses indivíduos e avaliando o resultado deles ao longo de um número definido de gerações, de acordo com a Figura 26.

Tabela 6. Variáveis utilizadas para a Otimização Multiobjetivo dos Modelos, onde n_entrada representa o número de neurônios de entrada.

Variável	Definição	Objetivo
Janela de Observação	n = n_entrada	Minimizar
Erro médio quadrático	$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$	Minimizar

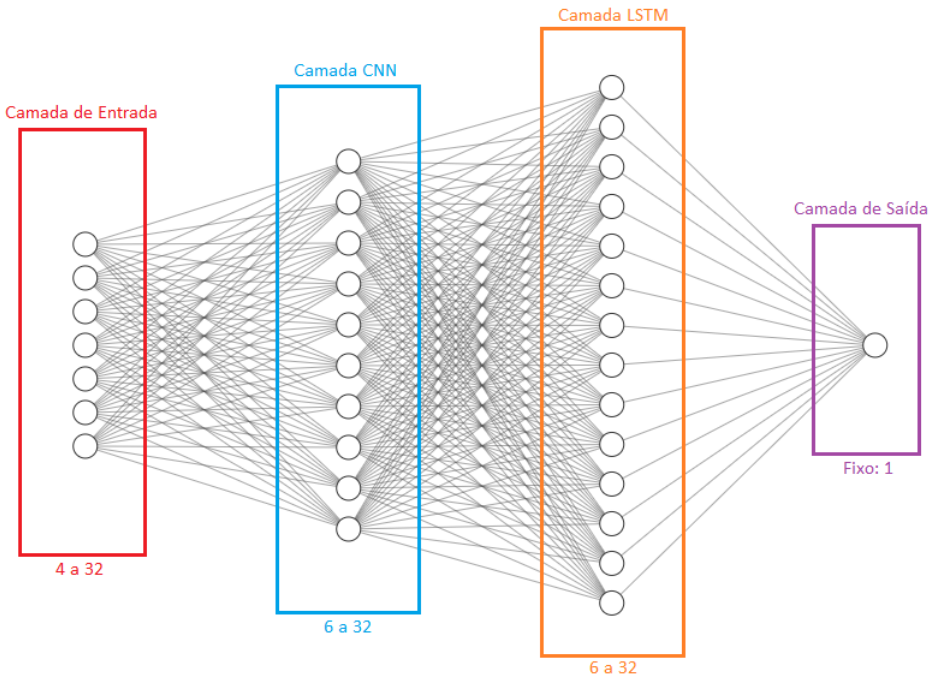
Figura 26. Diagrama de Blocos do processo de otimização dos modelos candidatos com NSGA-II



Fonte: Autor

Como demonstrado no diagrama presente na Figura 26, a otimização começa com a definição do indivíduo, que é um modelo candidato com atributos que correspondem a um determinado número de neurônios na camada de entrada, na camada CNN e na camada LSTM, como exposto na Figura 27. Após a definição do indivíduo, a população é inicializada aleatóriamente com vinte indivíduos com diferentes valores para seus atributos, sendo essa a geração inicial, na qual são aplicadas o crossover e a mutação com probabilidades especificadas para gerar novas soluções candidatas. A probabilidade de crossover é de setenta porcento e a probabilidade de mutação é de vinte porcento.

Figura 27. Representação simplificada dos indivíduos gerados no processo de otimização.



Fonte: Autor

Em seguida é realizado o treinamento da população de modelos candidatos usando o conjunto de dados de treinamento, e em seguida a avaliação multiobjetivo é realizada. Os candidatos são avaliados com base nos objetivos definidos, que são minimizar a janela de observação e o Erro Médio Quadrático no conjunto de teste, e as soluções encontradas são analisadas sobre o critério de não dominância. Em seguida, vinte indivíduos são selecionados da população atual para compor a presente geração, e então o processo continua por várias gerações até que a última geração seja atingida.

5 Resultados

5.1 Análise do Processo de Otimização

A saída resultante ao processo de otimização pode ser verificada na Tabela 7, que corresponde aos resultados do processo de otimização multiobjetivo conduzido pelo algoritmo NSGA-II. Cada linha na tabela representa uma iteração, portanto uma nova geração, e é denotada como *n_gen* (número de gerações), indicando o progresso do algoritmo ao longo do tempo. O *n_eval* (número de avaliações de fitness) representa o total de avaliações de aptidão realizadas até a geração atual. Já o *n_nds* (número de fronteiras não dominadas) refere-se à quantidade de soluções não dominadas no conjunto de soluções, utilizando a dominação de Pareto como critério de ordenação. A dominação de Pareto classifica as soluções em frentes não dominadas, destacando a natureza competitiva dos objetivos de otimização.

Tabela 7. Evolução do processo de otimização NSGA-II ao longo das gerações.

n_gen	n_eval	n_nds	eps	indicator
1	20	6	-	-
2	40	5	0.0769230769	ideal
3	60	5	0.0261846951	f
4	80	4	0.1251542455	nadir
5	100	3	0.0638511534	f
6	120	3	0.0111181662	f
7	140	3	0.000000E+00	f
8	160	3	0.0039895929	f
9	180	3	0.0025824689	ideal
10	200	3	0.0330489078	ideal
11	220	3	0.0373463413	nadir
12	240	3	0.000000E+00	f
13	260	3	0.000000E+00	f
14	280	3	0.000000E+00	f
15	300	3	0.000000E+00	f
16	320	3	0.000000E+00	f
17	340	3	0.000000E+00	f
18	360	3	0.000000E+00	f
19	380	3	0.000000E+00	f
20	400	3	0.000000E+00	f

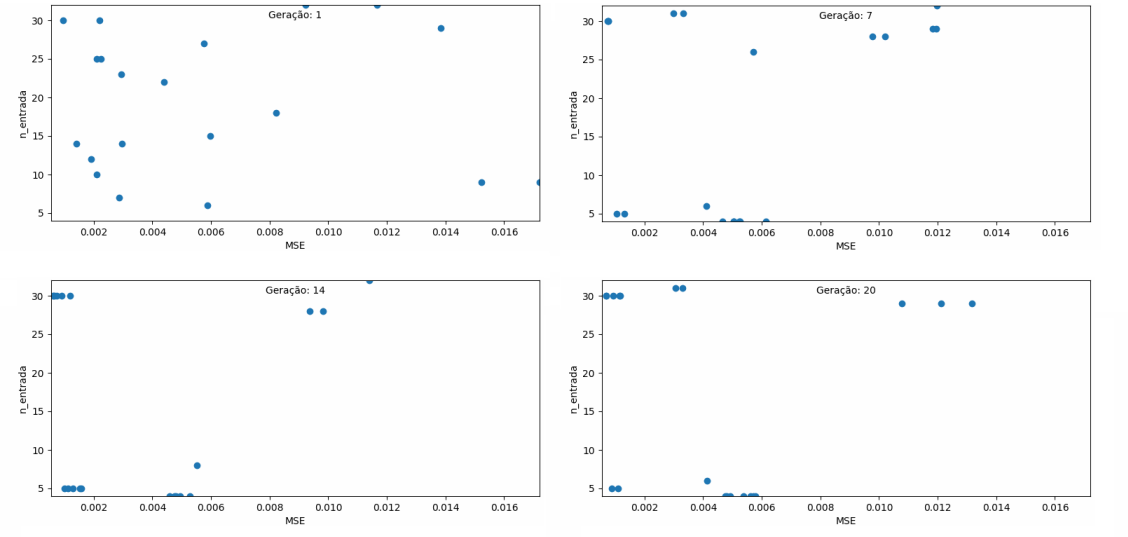
Ainda analisando a saída presente em Tabela 7 nota-se o parâmetro *eps*, que é associado ao hipervolume, uma métrica que quantifica a qualidade da frente de Pareto.

Valores mais elevados indicam uma maior distância das soluções não dominadas em relação ao ponto de referência ideal. Este ponto de referência é determinado pelos objetivos do problema, e a maximização do hipervolume é frequentemente um objetivo desejável. Finalmente, o último campo, *indicator* fornece informações sobre a qualidade da frente de Pareto com base em diferentes métricas, como *ideal* ou *nadir*, na qual, quando o indicador é designado como *ideal*, isso sugere que a frente de Pareto está se aproximando do ponto ideal em termos dos objetivos de otimização. Em outras palavras, as soluções não dominadas na frente de Pareto estão convergindo em direção à solução ótima ideal para o problema. Maximizar o indicador *ideal* é geralmente considerado desejável, indicando uma melhoria na qualidade global das soluções.

Por outro lado, quando o indicador é especificado como *nadir*, refere-se ao ponto nadir na frente de Pareto. O ponto nadir representa as piores soluções em cada objetivo na frente de Pareto. Minimizar a distância até o ponto nadir é uma indicação de que as soluções estão melhorando em relação às piores soluções conhecidas. Neste caso, minimizar o indicador *nadir* é geralmente considerado um objetivo positivo. Além das duas categorias mencionadas, pode haver outros indicadores específicos, como no exemplo dado (f). Esses indicadores adicionais podem representar critérios específicos de avaliação que não se enquadram nas categorias *ideal* ou *nadir*, e no presente caso, refere-se à função de objetivo associada à minimização dos objetivos do problema anteriormente citadas na Tabela 6. Em algoritmos de otimização multiobjetivo, que representam as métricas que o algoritmo busca minimizar.

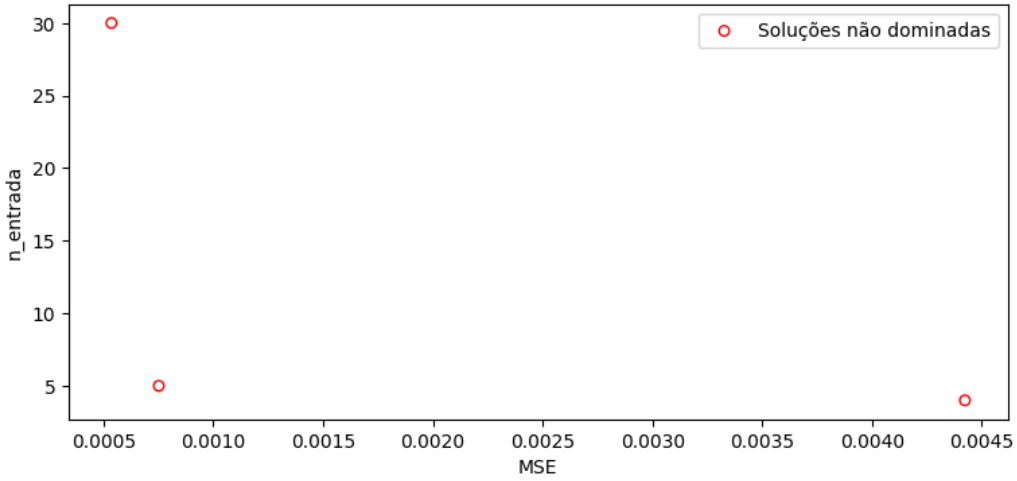
Observando a progressão desses resultados ao longo das iterações, observa-se que as gerações iniciais do algoritmo parecem priorizar a diversidade, conforme indicado pelo aumento inicial no número de fronteiras não dominadas, indicando uma maior exploração do espaço de busca. Posteriormente, ocorre uma convergência em direção a soluções ideais, evidenciada pelo aumento do hipervolume e pela Figura 28, que se aproxima do valor ideal. Nota-se que a métrica do hipervolume não apresenta variação significativa a partir da geração doze, o que indica que houve um número de gerações suficiente para a convergência das soluções não dominadas, e ao final do processo de otimização foram avaliados um total de quatrocentos modelos com diferentes topologias, sendo vinte modelos a cada uma das vintes gerações. As soluções finais podem ser visualizadas na Figura 29.

Figura 28. Avaliação dos indivíduos presentes nas gerações 1, 7, 14 e 20



Fonte: Autor

Figura 29. Soluções não dominadas ao final da otimização



Fonte: Autor

5.2 Avaliação dos modelos encontrados

A Tabela 8 apresenta os indivíduos resultantes do processo de otimização apresentado, que visou avaliar diferentes modelos, cada um caracterizado por configurações específicas de neurônios nas camadas de entrada, CNN e LSTM. O objetivo desse processo foi minimizar tanto o número de neurônios na camada de entrada quanto o Erro Médio

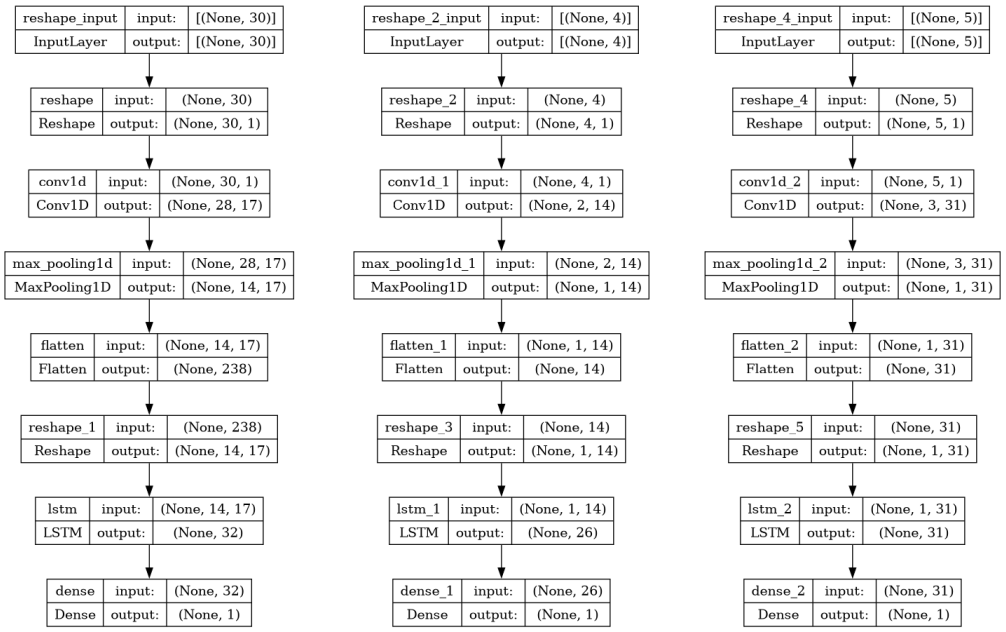
Quadrático (MSE) no conjunto de teste, que foi o parâmetro utilizado para avaliar a capacidade de aferição dos modelos, e as três soluções não dominadas encontradas representam os melhores modelos candidatos, denominados Modelo 1, 2 e 3, descritos na presente Tabela.

Tabela 8. Indivíduos gerados pelo processo de otimização.

Nome Atribuído	Entrada	Camada CNN	Camada LSTM	MSE
Modelo 1	30	17	32	0.0005357632180675864
Modelo 2	4	14	26	0.004422813653945923
Modelo 3	5	31	31	0.0007514497265219688

Analisando os resultados individualmente, o Modelo 1 destaca-se alcançando o menor MSE no conjunto de teste dentre os todos os modelos, porque o mesmo possui a maior janela de observação. Por sua vez, o Modelo 2 apresenta uma configuração com o menor número de neurônios na camada de entrada, porém observa-se o maior MSE dentre as soluções não dominadas. O Modelo 3 por sua vez possui um neurônio a mais na camada de entrada em relação ao Modelo 2, e um MSE no conjunto de teste bem menor que o do mesmo, mas ainda maior que o do Modelo 1. A interpretação desses resultados é realizada comparando o desempenho dos modelos em relação ao número de neurônios na camada de entrada e ao MSE. A preferência recai sobre modelos que minimizem ambas as métricas, destacando a importância de equilibrar a complexidade do modelo com a capacidade de generalização. A topologia final dos três modelos pode ser verificada na Figura 30.

Figura 30. Topologia final do Modelo 1 (à esquerda), Modelo 2 (centro) e Modelo 3 (à direita)



Fonte: Autor

Em termos gerais, a escolha do melhor modelo é uma ponderação entre a complexidade do modelo, portanto, o número de neurônios na camada de entrada, e a capacidade de generalização, correspondente ao MSE no conjunto de teste. Ressalta-se também que o tamanho da janela de observação tornar-se-á um tempo de observação ao processo de descarga, que no presente estudo é de um minuto por ponto na mesma, tornando janelas de observação maiores mais difíceis de implementar em uma aplicação para gerenciar um evento em tempo real.

A Tabela 9 lista a relação do tempo necessário para a janela de observação de cada modelo e o seu MSE aplicado à escala de tempo analisada. Nota-se que o Modelo mais preciso necessita de um total de trinta minutos observando uma curva de descarga para fazer a estimativa do tempo de descarga, o que é um intervalo de tempo considerável apesar da escala do experimento se estender por mais de setenta horas. Já o modelo com a menor janela necessita apenas de quatro minutos para realizar uma estimativa, porém o erro médio de estimativa é muito maior que o modelo mais preciso. O último modelo, no entanto, necessita de cinco minutos para sua aferição, e sua diferença de precisão em relação ao primeiro modelo é de menos de um minuto.

Tabela 9. Comparação do tempo de observação necessário e MSE de cada modelo.

Nome Atribuído	Tempo da Janela de Observação	MSE do tempo de descarga
Modelo 1	30 min	∼2 min e 18 s
Modelo 2	4 min	∼19 min e 3 s
Modelo 3	5 min	∼3 min e 14 s
Tempo máximo da escala do experimento:		∼71 h, 46 min e 37 s

6 Conclusão

A realização deste trabalho proporcionou uma abordagem completa desde a idealização do cenário proposto para a Rede de Sensores IoT, até a implementação de um modelo de estimação do tempo de descarga das baterias desses dispositivos. Foram realizados com sucesso testes congruentes com o cenário proposto, envolvendo uma rede IoT com diferentes tipos de sensores, alimentados por baterias recarregáveis e conectados por uma rede LPWAN, tornando a aquisição de dados bem sucedida.

A metodologia de treinamento do modelo de estimação do tempo de descarga, baseada em uma janela de observação e com o uso da otimização multiobjetivo utilizando o algoritmo NSGA-II, apresentou uma boa convergência em relação aos resultados considerados ótimos para o contexto apresentado. Ao definir o processo de otimização, visando encontrar modelos que minimizem tanto o número de neurônios na camada de entrada, quanto o erro médio quadrático no conjunto de teste, proporcionou-se uma forma dinâmica de testar várias topologias de forma ordenada e validar as mesmas durante o processo, resultando em dois modelos que apresentaram notável capacidade de generalização, atingindo o objetivo proposto de realizar estimativas do tempo de vida útil restante ao dispositivo observando o progresso da curva de descarga com a margem de erro de poucos minutos.

Em conclusão, este trabalho demonstra que é possível realizar uma abordagem abrangente e generalista, para estimação do tempo de vida útil de um dispositivo IoT, dentro do cenário especificado, proporcionando estimativas sem nenhuma informação técnica direta dos dispositivos e sensores envolvidos, o que torna propício a aferição de qualquer dispositivo que se encaixe nas mesmas condições analíticas do experimento realizado através do método desenvolvido, o que caracteriza um passo significativo para o desenvolvimento de redes IoT mais inteligentes e auto gerenciáveis.

6.1 Trabalhos Futuros

Pretende-se realizar novos experimentos de descarga de dispositivos, levando-se em consideração outros fatores que afetam o consumo tais como, parâmetros da propagação do sinal do transmissor LoRa, bem como realização de testes com mais sensores de diferentes naturezas para criação de uma base de dados mais abrangente objetivando o aprendizado dos modelos. Outra vertente, seguinte aos resultados deste estudo, é o desenvolvimento de modelos mais precisos e que abranjam outros aspectos da rede de sensores com o objetivo do desenvolvimento de simuladores precisos para realização de outros tipos de previsão, ou mesmo *Digital Twins* para realizar testes virtuais em tempo real com dados reais coletados pela rede IoT.

Referências

ABINC. *IoT em 2024: novas tendências apontam para uma revolução conectada*. 2023. Disponível em: <https://abinc.org.br/iot-em-2024-novas-tendencias-apontam-para-uma-revolucao-conectada/>. Acesso em: 24 dec 2023. Citado na página 16.

ALBEYBONI, H. J.; ALI, I. A. Lpwan technologies for iot applications: A review. *Journal of Duhok University*, v. 26, n. 1, p. 29–42, 2023. Citado 2 vezes nas páginas 29 e 30.

ALVES, G. *Understanding ConvNets (CNN)*. 2018. Disponível em: <https://medium.com/neuronio/understanding-convnets-cnn-712f2afe4dd3>. Acesso em: 12 jan 2024. Citado na página 36.

AMAZON WEB SERVICES, INC. *O que é RNN?* 2023. Disponível em: <https://aws.amazon.com/pt/what-is/recurrent-neural-network/>. Acesso em: 12 jan 2024. Citado na página 36.

ANALOG DEVICES, INC. *Sensor*. 2024. Disponível em: <https://www.analog.com/en/resources/glossary/sensor.html>. Acesso em: 10 jan 2024. Citado na página 25.

APOGEEWEB. *Sensor Basics: Types of Sensors*. 2018. Disponível em: <https://www.apogeeweb.net/article/91.html>. Acesso em: 10 jan 2024. Citado na página 25.

ASCENTY. *A Inteligência Artificial na Era da Internet das Coisas*. 2023. Disponível em: <https://ascenty.com/blog/artigos/a-inteligencia-artificial-na-era-da-internet-das-coisas/>. Acesso em: 24 dec 2023. Citado na página 16.

CARVALHO, A. P. d. L. F. d. *Perceptron Multi-Camadas (MLP)*. 2009. Disponível em: <https://sites.icmc.usp.br/andre/research/neural/mlp.htm>. Acesso em: 05 jan 2024. Citado na página 33.

CHANDRAN, V. e. a. State of charge estimation of lithium-ion battery for electric vehicles using machine learning algorithms. *World Electric Vehicle Journal*, v. 12, n. 1, p. 38, 2021. Citado 2 vezes nas páginas 21 e 22.

DATA SCIENCE ACADEMY. *Capítulo 40 – Introdução às Redes Neurais Convolucionais*. 2022. Disponível em: <https://www.deeplearningbook.com.br/introducao-as-redes-neurais-convolucionais/#:~:text=Uma%20Rede%20Neural%20Convolucional%20(ConvNet,de%20diferenciar%20um%20do%20outro.> Acesso em: 12 jan 2024. Citado na página 35.

DEB, K. e. a. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, v. 6, n. 2, p. 182–197, 2002. Citado na página 38.

ELECTRONICSHUB. *Basics of Microcontrollers – History, Structure and Applications*. 2017. Disponível em: <https://www.electronicshub.org/microcontrollers-basics-structure-applications/>. Acesso em: 10 jan 2024. Citado na página 26.

FAHMIDEH, M.; ZOWGHI, D. An exploration of iot platform development. *Information Systems*, v. 87, p. 101409, 2020. Citado na página 27.

GOOGLE. *What is Deep Learning?* 2024. Disponível em: <https://cloud.google.com/discover/what-is-deep-learning>. Acesso em: 03 jan 2024. Citado na página 35.

IBRAHIM, D. *Microcontroller Based Applied Digital Control*. [S.l.]: John Wiley, 2006. Citado na página 26.

INPIXON. *What is Chirp (CSS) Technology?* 2024. Disponível em: <https://www.inpixon.com/technology/standards/chirp-spread-spectrum>. Acesso em: 19 fev 2024. Citado na página 31.

INTERNATIONAL BUSINESS MACHINES CORPORATION. *O que são redes neurais convolucionais?* 2024. Disponível em: <https://www.ibm.com/br-pt/topics/convolutional-neural-networks>. Acesso em: 12 jan 2024. Citado na página 35.

JANIESCH, C.; ZSCHECH, P.; HEINRICH, K. Machine learning and deep learning. *Electronic Markets*, v. 31, n. 3, p. 685–695, 2021. Citado na página 34.

JENKINS, I.; GEE, L.; KNAUSS, A.; YIN, H.; SCHROEDER, J. Accident scenario generation with recurrent neural networks. In: . [S.l.: s.n.], 2018. p. 3340–3345. Citado na página 38.

JOSTJUN, D. *What is a sensor?* 2019. Disponível em: <https://www.fierceelectronics.com/sensors/what-a-sensor>. Acesso em: 10 jan 2024. Citado na página 25.

KALIRANE, M. *Gradient Descent vs. Backpropagation: What's the Difference?* 2023. Disponível em: <https://www.analyticsvidhya.com/blog/2023/01/gradient-descent-vs-backpropagation-whats-the-difference/>. Acesso em: 04 jan 2024. Citado na página 33.

KHAN, M.; SHOAIB, M.; HAMMAD, M.; SALAHUDIN, H.; AHMAD, F.; AHMAD, S. Application of machine learning techniques in rainfall–runoff modelling of the soan river basin, pakistan. *Water*, v. 13, p. 3528, 12 2021. Citado na página 33.

KRENKER, A.; BEŠTER, J.; KOS, A. Introduction to the artificial neural networks. In: *Artificial Neural Networks: Methodological Advances and Biomedical Applications*. [S.l.]: InTech, 2011. p. 1–18. Citado na página 32.

LI, Y.; YANG, J.; WANG, J. Dylora: Towards energy efficient dynamic lora transmission control. In: *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. [S.l.]: IEEE, 2020. p. 2312–2320. Citado 2 vezes nas páginas 20 e 22.

LORA ALLIANCE. *What is LoRaWAN® Specification*. 2024. Disponível em: <https://lora-alliance.org/about-lorawan/>. Acesso em: 02 jan 2024. Citado na página 31.

LOUREIRO, A. A. e. a. Redes de sensores sem fio. *Simpósio Brasileiro de Redes de Computadores (SBRC)*, p. 179–226, 2003. Citado 2 vezes nas páginas 24 e 25.

MONTEIRO, J. *IoT deve crescer a quase 20% por ano no Brasil até 2027, aponta estudo*. 2023. Disponível em: <https://ipnews.com.br/iot-deve-crescer-a-quase-20-por-ano-no-brasil-ate-2027-aponta-estudo/>. Acesso em: 24 dec 2023. Citado na página 16.

NURGALIYEV, M. e. a. Prediction of energy consumption for lora based wireless sensors network. *Wireless Networks*, v. 26, p. 3507–3520, 2020. Citado 2 vezes nas páginas 20 e 22.

OGUNSEYE, T.; EGBEYALE, G.; BELLO, A.; AJANI, S. *Design and Construction of a Microcontroller Based Electronic Moving Message Display*. 2021. Citado na página 26.

PADRó, F.; OZóN, J.; APLICADA, D. Graph coloring algorithms for assignment problems in radio networks. 08 1995. Citado na página 39.

PEREIRA, M. R.; AMORIM, C. L. D.; CASTRO, M. C. S. D. Tutorial sobre redes de sensores. *Cadernos do IME-Série Informática*, v. 14, n. 14, p. 39–53, 2003. Citado 2 vezes nas páginas 24 e 26.

POPESCU, M.-C. e. a. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, v. 8, n. 7, p. 579–588, 2009. Citado na página 32.

PYTHON SOFTWARE FOUNDATION. *What is Python? Executive Summary*. 2024. Disponível em: <https://www.python.org/doc/essays/blurb/>. Acesso em: 14 jan 2024. Citado na página 34.

PÉREZ, M. A. F. *Problemas de Otimização Multiobjetivo*. 2012. Disponível em: <https://www.maxwell.vrac.puc-rio.br/19601/19601_4.PDF>. Acesso em: 12 jan 2024. Citado na página 38.

RAJAB, H. e. a. Evaluation of energy consumption of lpwan technologies. *EURASIP Journal on Wireless Communications and Networking*, v. 2023, n. 1, p. 118, 2023. Citado 3 vezes nas páginas 21, 22 e 23.

RAKWIRELESS TECHNOLOGY LIMITED. *RAK4630 WisBlock LPWAN+BLE Module Datasheet*. 2024. Disponível em: <https://docs.rakwireless.com/Product-Categories/WisDuo/RAK4630-Module/Datasheet/>. Acesso em: 06 jan 2024. Citado na página 43.

ROSSATO, J.; SPANHOL, F. A.; CAMARGO, E. T. D. Implantação e avaliação de uma rede sem-fio de longo alcance e baixa potência para cidades inteligentes. p. 192–205, 2020. Citado na página 29.

SANTOS, M. M. e. a. Internet das coisas: a busca do conceito e as perspectivas futuras sobre sua aplicabilidade. *Research, Society and Development*, v. 10, n. 10, p. e140101018504–e140101018504, 2021. Citado na página 27.

SARKER, I. H. Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions. *SN Computer Science*, v. 2, n. 6, p. 420, 2021. Citado na página 34.

SEMTECH. *What are LoRa® and LoRaWAN®?* 2024. Disponível em: <https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan/>. Acesso em: 02 jan 2024. Citado na página 30.

SHEN, S. e. a. A deep learning method for online capacity estimation of lithium-ion batteries. *Journal of Energy Storage*, v. 25, p. 100817, 2019. Citado 2 vezes nas páginas 20 e 22.

SILVA, G. S. d. e. a. *Avaliação de arquiteturas de Redes Neurais Recorrentes para Reconhecimento de Atividades Humanas utilizando acelerômetros de dispositivos móveis*. 2022. Citado na página 36.

SINGH, R. K. e. a. Energy consumption analysis of lpwan technologies and lifetime estimation for iot application. *Sensors*, v. 20, n. 17, p. 4794, 2020. Citado na página 29.

SIQUEIRA, H. V. *Metaheurísticas de Otimização Bio-Inspiradas*. 2019. Disponível em: <http://paginapessoal.utfpr.edu.br/hugosiqueira/disciplinas-mestrado/metaheuristicas-de-otimizacao-bio-inspiradas/Aula%2019%20-%20Multiobjetivo.pdf>. Acesso em: 12 jan 2024. Citado na página 38.

SOARES, A. *Redes Neurais Profundas – Deep Learning*. 2018. Disponível em: <https://ww2.inf.ufg.br/~anderson/deeplearning/20181/Aula%20-%20Redes%20Neurais%20Convolucionais%20Parte%20I.pdf>. Acesso em: 12 jan 2024. Citado na página 35.

SU, B.; QIN, Z.; NI, Q. Energy efficient uplink transmissions in lora networks. *IEEE Transactions on Communications*, v. 68, n. 8, p. 4960–4972, 2020. Citado 2 vezes nas páginas 20 e 22.

TAMILSELVI, S. e. a. A review on battery modelling techniques. *Sustainability*, v. 13, n. 18, p. 10042, 2021. Citado 2 vezes nas páginas 21 e 22.

THE MATHWORKS, INC. *ThingSpeak Documentation*. 2024. Disponível em: <https://www.mathworks.com/help/thingspeak/>. Acesso em: 03 jan 2024. Citado na página 28.

THE THINGS INDUSTRIES. *What are LoRa and LoRaWAN?* 2024. Disponível em: <https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/>. Acesso em: 02 jan 2024. Citado 2 vezes nas páginas 30 e 31.

THOMAZINI, D.; ALBUQUERQUE, P. U. B. D. *Sensores industriais: fundamentos e aplicações*. 9. ed. São Paulo: Saraiva Educação SA, 2020. Citado na página 24.

TONDAK, A. *Recurrent Neural Networks (RNN) Tutorial*. 2023. Disponível em: <https://k21academy.com/datascience-blog/machine-learning/recurrent-neural-networks/>. Acesso em: 12 jan 2024. Citado na página 37.

TREND MICRO INCORPORATED. *LoRaWAN*. 2024. Disponível em: <https://www.trendmicro.com/vinfo/us/security/definition/lorawan>. Acesso em: 02 jan 2024. Citado na página 30.

USMANI, Z.-U.-H. *What is Kaggle, Why I Participate, What is the Impact?* 2017. Disponível em: <https://www.kaggle.com/discussions/getting-started/44916>. Acesso em: 14 jan 2024. Citado na página 34.

VASCO, L. P. *Um estudo de redes neurais recorrentes no contexto de previsões no mercado financeiro*. 2020. Citado na página 37.

VÄÄNÄNEN, O.; HÄMÄLÄINEN, T. Lora-based sensor node energy consumption with data compression. In: *2021 IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4.0&IoT)*. [S.l.]: IEEE, 2021. p. 6–11. Citado 3 vezes nas páginas 21, 22 e 23.

WALCZAK, S. Artificial neural networks. In: _____. *Advanced Methodologies and Technologies in Artificial Intelligence, Computer Simulation, and Human-Computer Interaction*. [S.l.]: IGI Global, 2019. p. 40–53. Citado na página 32.

YEFREMOV, Y. *TTGO T-Beam*. 2024. Disponível em: <https://doc.riot-os.org/group__boards___esp32___ttgo-t-beam.html>. Acesso em: 07 jan 2024. Citado na página 43.

ZABEU, S. *NB-IoT e LoRaWAN dominarão o universo LPWAN até 2028*. 2023. Disponível em: <https://network-king.net/pt-pt/nb-iot-e-lorawan-dominarao-o-universo-lpwan-ate-2028/>. Acesso em: 24 dec 2023. Citado na página 16.

ZIKRIA, Y. B. e. a. Next-generation internet of things (iot): Opportunities, challenges, and solutions. *Sensors*, v. 21, n. 4, p. 1174, 2021. Citado na página 27.

Apêndices

APÊNDICE A . Algoritmo do Projeto

O script presente na página seguinte é responsável por extrair os dados tabulares de arquivos extraídos da API ThingSpeak, separar as curvas de descarga de cada experimento, construir o ambiente para aplicação da otimização multiobjetivo e realizar a otimização registrando os melhores resultados ao final das iterações, ressaltando que o processo de validação com o banco de dados de teste já ocorre durante o processo de otimização.

O código foi desenvolvido em Python, e foi executado em um ambiente Jupyter de uma Virtual Machine do Kaggle, permitindo que todas as saídas registradas na execução do projeto fossem registradas no corpo do arquivo. O presente script e todas as bases de dados utilizadas podem ser acessadas no repositório oficial do projeto, disponível em https://github.com/WCosmo/PPGEE-Estimacao-De-Descarga.

Inicialização

Resgistro do tempo de execução:

```
import time
start_time = time.time()
```

Instalação da biblioteca para NSGA-II:

```
!pip install -q pymoo
```

Inicialização das bibliotecas utilizadas:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Bibliotecas para as redes Deep Learning:

```
import tensorflow as tf
from tensorflow import keras
from keras import layers
from keras import models
from keras import backend
from tensorflow.keras.utils import plot_model
from sklearn.model_selection import train_test_split

keras.utils.set_random_seed(6)
```

```
2024-02-07 04:12:40.095609: E
external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261] Unable
to register cuDNN factory: Attempting to register factory for plugin
cuDNN when one has already been registered
2024-02-07 04:12:40.095723: E
external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Unable to
register cuFFT factory: Attempting to register factory for plugin
cuFFT when one has already been registered
2024-02-07 04:12:40.223346: E
external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515] Unable
to register cuBLAS factory: Attempting to register factory for plugin
cuBLAS when one has already been registered
```

Inicialização do NSGA-II:

```
from pymoo.core.problem import Problem
from pymoo.algorithms.moo.nsga2 import NSGA2
```

```
from pymoo.optimize import minimize
from pymoo.operators.mutation.pm import PolynomialMutation
from pymoo.operators.crossover.sbx import  SBX
from pymoo.operators.repair.rounding import RoundingRepair
from pymoo.operators.sampling.rnd import IntegerRandomSampling
from pymoo.core.population import Population
```

```
2024-02-07 04:12:52,249    INFO util.py:124 -- Outdated packages:
  ipywidgets==7.7.1 found, needs ipywidgets>=8
Run `pip install -U ipywidgets`, then restart the notebook server for
rich notebook output.
```

Carregamento dos datasets:

```
data1 = pd.read_csv('https://raw.githubusercontent.com/WCosmo/PPGEE-
Projeto-de-mestrado/main/base_de_dados/thingspeak_canal_01.csv')
data2 = pd.read_csv('https://raw.githubusercontent.com/WCosmo/PPGEE-
Projeto-de-mestrado/main/base_de_dados/thingspeak_canal_02.csv')
data3 = pd.read_csv('https://raw.githubusercontent.com/WCosmo/PPGEE-
Projeto-de-mestrado/main/base_de_dados/thingspeak_canal_03.csv')
```

Pré processamento dos dados

Cada curva de descarga está associada a uma "tag" na API da Thingspeak, a seguir divide-se os dados separando as curvas em vetores:

```
timestamp_e1 = []
timestamp_e2 = []
timestamp_e3 = []
timestamp_e4 = []
timestamp_e5 = []

d_curve_e1 = []
d_curve_e2 = []
d_curve_e3 = []
d_curve_e4 = []
d_curve_e5 = []

for i in range(len(data2['field1'].to_numpy())):
  if data2['field2'].to_numpy()[i] < 3.7 and
data2['field2'].to_numpy()[i] > 2.70 :
    if data2['field1'].to_numpy()[i] == 1: #Retornar apenas o
experimento 1
      timestamp_e1.append(data2['created_at'][i])
      d_curve_e1.append(data2['field2'].to_numpy()[i])
    if data2['field1'].to_numpy()[i] == 2: #Retornar apenas o
experimento 3
```

```
      timestamp_e3.append(data2['created_at'][i])
      d_curve_e3.append(data2['field2'].to_numpy()[i])

for i in range(len(data1['field1'].to_numpy())):
  if data1['field2'].to_numpy()[i] < 3.7 and
data1['field2'].to_numpy()[i] > 2.70 :
    if data1['field1'].to_numpy()[i] == 2: #Retornar apenas o
experimento 2
      timestamp_e2.append(data1['created_at'][i])
      d_curve_e2.append(data1['field2'].to_numpy()[i])

for i in range(len(data3['field1'].to_numpy())):
  if data3['field2'].to_numpy()[i] < 3.7 and
data3['field2'].to_numpy()[i] > 2.70 :
    if data3['field1'].to_numpy()[i] == 0: #Retornar apenas o
experimento 4
      timestamp_e4.append(data3['created_at'][i])
      d_curve_e4.append(data3['field2'].to_numpy()[i])
    if data3['field1'].to_numpy()[i] == 1: #Retornar apenas o
experimento 5
      timestamp_e5.append(data3['created_at'][i])
      d_curve_e5.append(data3['field2'].to_numpy()[i])

d_curve_e1 = np.array(d_curve_e1)
d_curve_e2 = np.array(d_curve_e2)
d_curve_e3 = np.array(d_curve_e3[2:])
d_curve_e4 = np.array(d_curve_e4)
d_curve_e5 = np.array(d_curve_e5)
```

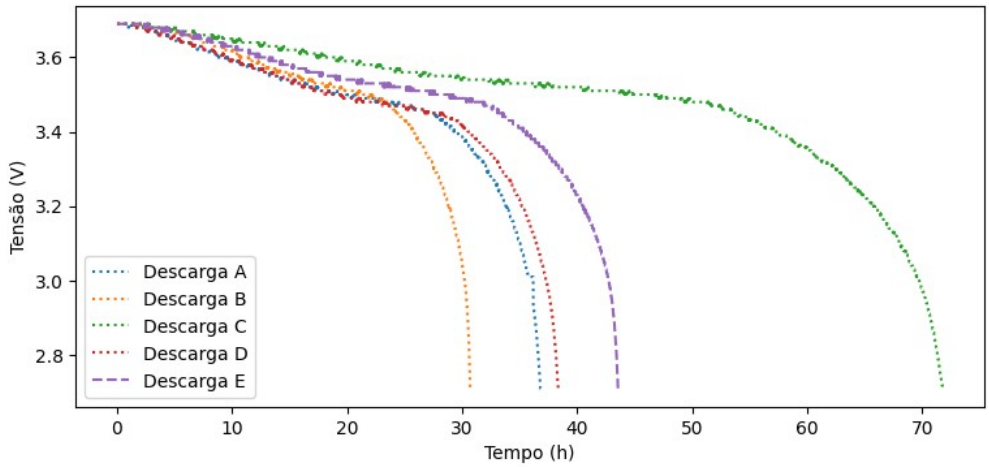
Função para transformar o timestamp no formato "yyyy/mm/dd-hh:mm:ss" para apenas segundo, que é a menor unidade de tempo presente no timestamp:

```
def timestamper(timestamp):

  ft = 0
  ii = 0
  timescale = []

  for t in timestamp:
    sbuff = t.split('T')
    s_ymd = sbuff[0].split('-')
    ymd = int(s_ymd[0])*31536000 + int(s_ymd[1])*2592000 +
int(s_ymd[2])*86400 #converter yyyy/mm/dd para s

    bs_hr = sbuff[1].split('-')
    s_hr = bs_hr[0].split(':')
    hr = int(s_hr[0])*3600 + int(s_hr[1])*60 + int(s_hr[2]) #converter
hh:mm:ss para s
```

```
    ts = ymd + hr
    timescale.append(ts)


  timescale = np.array(timescale)
  return timescale - min(timescale)

t_e1 = timestamper(timestamp_e1[:])
t_e2 = timestamper(timestamp_e2[:])
t_e3 = timestamper(timestamp_e3[2:])
t_e4 = timestamper(timestamp_e4[:])
t_e5 = timestamper(timestamp_e5[:])
```

Plot das curvas de descarga com a escala de tempo correspondente:

```
plt.rcParams['figure.figsize'] = (9, 4)
plt.plot(t_e1/(60*60),d_curve_e1, ls = ':', label='Descarga
A')#DS18B20
plt.plot(t_e2/(60*60),d_curve_e2, ls = ':', label='Descarga
B')#DS18B20 e Voltímetro
plt.plot(t_e4/(60*60),d_curve_e4, ls = ':', label='Descarga
C')#Termopar I2C
plt.plot(t_e3/(60*60),d_curve_e3, ls = ':', label='Descarga
D')#Voltímetro
plt.plot(t_e5/(60*60),d_curve_e5, ls = '--', label='Descarga
E')#Termopar I2C e INA219

plt.legend(loc='lower left')
#plt.title('Curvas de Descarga')
plt.xlabel('Tempo (h)')
plt.ylabel('Tensão (V)')
plt.show()
```



Função que gera instâncias a partir de um número de pontos (tamantnho do segmento observado):

```
def gen_dataset(curve, ts, pn):
  ll = len(curve)

  X = np.zeros((ll-pn, pn))
  Y = np.zeros(ll-pn)

  for i in range(ll-pn):
    for j in range(pn):
      X[i][pn-j-1] = curve[pn+i-j-1]


  Y = ts[pn:]
  Y = max(Y) - Y

  return (X,Y)
```

Outras funções pertinentes (normalização, reescalagem)

```
def norm(v, mi, ma):
  n_v = (v-mi)/(ma-mi)

  return n_v

def de_norm(n_v, mi, ma):
  v = ((n_v)*(ma-mi))+mi

  return v
```

Limiares da normalização dos dados:

```
X_min = 2.70
X_max = 3.70

print('Limite mínimo da tensão para normalização: ', X_min * 1000,
'mV')
print('Limite máximo da tensão para normalização: ', X_max * 1000,
'mV')

Y_min = min(np.concatenate((t_e1,t_e2,t_e3,t_e4,t_e5)))
Y_max = max(np.concatenate((t_e1,t_e2,t_e3,t_e4,t_e5)))

print('\nLimite mínimo do tempo para normalização: ', Y_min, 's')
print('Limite máximo do tempo para normalização: ', Y_max, 's')
```

```
Limite mínimo da tensão para normalização:  2700.0 mV
Limite máximo da tensão para normalização:  3700.0 mV

Limite mínimo do tempo para normalização:  0 s
Limite máximo do tempo para normalização:  258393 s
```

Número de épocas de treinamento do modelo:

```
ep = 10
```

Implementação da Otimização Multiobjetivo

Inicialização do ambiente GPU:

```
print("N. GPUs Disponíveis: ",
len(tf.config.list_physical_devices('GPU')))
```

```
N. GPUs Disponíveis:  1
```

```
physical_devices = tf.config.list_physical_devices('GPU')
tf.config.set_visible_devices(physical_devices[0], 'GPU')
```

Função para geração de Datasets de treino e teste baseados em um dado tamanho da janela de observação (número de neurônios de entrada):

```
def redmis_dataset(ne):

    d1 = gen_dataset(norm(d_curve_e1, X_min, X_max), norm(t_e1, Y_min,
Y_max), ne) #curva de descarga A
    d2 = gen_dataset(norm(d_curve_e3, X_min, X_max), norm(t_e3, Y_min,
Y_max), ne) #curva de descarga B
    d3 = gen_dataset(norm(d_curve_e2, X_min, X_max), norm(t_e2, Y_min,
Y_max), ne) #curva de descarga C
```

```
    d4 = gen_dataset(norm(d_curve_e4, X_min, X_max), norm(t_e4, Y_min,
Y_max), ne) #curva de descarga D
    d5 = gen_dataset(norm(d_curve_e5, X_min, X_max), norm(t_e5, Y_min,
Y_max), ne) #curva de descarga E

    #20% dos dados para validação
    d1Xf, d1Xv, d1Yf, d1Yv = train_test_split(d1[0], d1[1],
test_size=0.2, random_state=6)
    d2Xf, d2Xv, d2Yf, d2Yv = train_test_split(d2[0], d2[1],
test_size=0.2, random_state=6)
    d3Xf, d3Xv, d3Yf, d3Yv = train_test_split(d3[0], d3[1],
test_size=0.2, random_state=6)
    d4Xf, d4Xv, d4Yf, d4Yv = train_test_split(d4[0], d4[1],
test_size=0.2, random_state=6)

    #Treino: Descargas A, B, C e D (80%)
    X_treino = pd.DataFrame(np.concatenate((d1Xf, d2Xf, d3Xf, d4Xf)))
    Y_treino = pd.DataFrame(np.concatenate((d1Yf, d2Yf, d3Yf, d4Yf)))

    #Validação: Descargas A, B, C e D (20%)
    X_val = pd.DataFrame(np.concatenate((d1Xv, d2Xv, d3Xv, d4Xv)))
    Y_val = pd.DataFrame(np.concatenate((d1Yv, d2Yv, d3Yv, d4Yv)))

    #Teste: Descargas E
    X_teste = pd.DataFrame(d5[0])
    Y_teste = pd.DataFrame(d5[1])

    return X_treino, Y_treino, X_teste, Y_teste, X_val, Y_val
```

Função para avaliação de um indivíduo. Um indivíduo consiste em um combinação de:

- Número de neurônios da camada de entrada (n_entrada)
- Número de neurônios na camada CNN (n_cnn)
- Número de neurônios da camada LSTM (n_lstm)

```
def evaluate_ind(individual):
    n_cnn, n_lstm, n_entrada = individual

    with tf.device('/GPU:0'):
      model = keras.models.Sequential()
      model.add(keras.layers.Reshape((n_entrada, 1),
input_shape=(n_entrada,)))

      #1st layer - CNN
      model.add(keras.layers.Conv1D(n_cnn, 3, activation='relu'))
      model.add(keras.layers.MaxPooling1D(2))

      #2nd layer - LSTM
      model.add(keras.layers.Flatten())
      model.add(keras.layers.Reshape((int((n_entrada-2)/2), n_cnn)))
```

```
      model.add(keras.layers.LSTM(n_lstm, activation='tanh'))

      #Output layer
      model.add(keras.layers.Dense(1, activation='sigmoid'))

      model.compile(optimizer='adam',
loss=keras.losses.MeanSquaredError(), metrics=['mse'])
      Xf, Yf, Xt, Yt, Xv, Yv = redmis_dataset(n_entrada)

      h = model.fit(Xf, Yf, epochs=ep, batch_size=64, verbose=0,
validation_data=(Xv, Yv))

      eval_mse = model.evaluate(Xt, Yt, verbose=0)

      keras.backend.clear_session()

    return eval_mse[1], n_entrada
```

Função para avaliar uma populaçao composta de indivíduos:

```
def evaluate_pop(population):
    n_cnn, n_lstm, n_entrada = population

    mse_fitness = []
    ne_fitness = []
    for n in range(len(n_cnn)):
        mse, ne = evaluate_ind((n_cnn[n], n_lstm[n], n_entrada[n]))
        mse_fitness.append(mse)
        ne_fitness.append(ne)

    return np.array(mse_fitness), np.array(ne_fitness)
```

Definição da classe do problema a ser otimizado:

```
class targetProblem(Problem):
    def __init__(self):
      super().__init__(n_var=3, n_obj=2, n_constr=0, xl=[6, 6, 4],
xu=[32, 32, 32], vtype=int)

    def _evaluate(self, x, out, *args, **kwargs):
        n_cnn = x[:,0]
        n_lstm = x[:,1]
        n_entrada = x[:,2]

        mse, _ = evaluate_pop((n_cnn, n_lstm, n_entrada))
        out["F"] = [mse, n_entrada]
```

Implementação do algoritmo NSGA2 no problema definido:

```
ps = 20
problem = targetProblem()
algorithm = NSGA2(
    pop_size=ps,
    sampling=IntegerRandomSampling(),
    crossover=SBX(prob=0.7, eta=20, vtype=float,
repair=RoundingRepair()),
    mutation=PolynomialMutation(prob=0.2, eta=20, vtype=float,
repair=RoundingRepair()),
    eliminate_duplicates=True
)

print('\n\nNSGA-II inicializado - tamanho da população: ', ps)
res = minimize(problem, algorithm, ('n_gen', 20), verbose=True,
save_history=True, seed=6)

best_top = res.X
best_sol = res.F
```

```
NSGA-II inicializado - tamanho da população:  20
```

```
WARNING: All log messages before absl::InitializeLog() is called are
written to STDERR
I0000 00:00:1707279179.736571      80 device_compiler.h:186] Compiled
cluster using XLA!  This line is logged at most once for the lifetime
of the process.
```

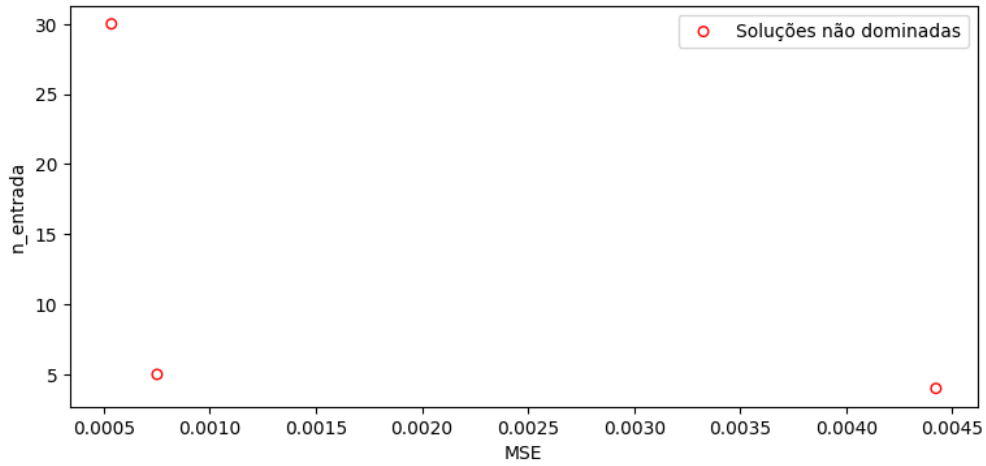
```
=========================================================
n_gen  |  n_eval  | n_nds  |      eps      |   indicator
=========================================================
     1 |       20 |      6 |             - |             -
     2 |       40 |      5 |  0.0769230769 |         ideal
     3 |       60 |      5 |  0.0261846951 |             f
     4 |       80 |      4 |  0.1251542455 |         nadir
     5 |      100 |      3 |  0.0638511534 |             f
     6 |      120 |      3 |  0.0111181662 |             f
     7 |      140 |      3 |  0.000000E+00 |             f
     8 |      160 |      3 |  0.0039895929 |             f
     9 |      180 |      3 |  0.0025824689 |         ideal
    10 |      200 |      3 |  0.0330489078 |         ideal
    11 |      220 |      3 |  0.0373463413 |         nadir
    12 |      240 |      3 |  0.000000E+00 |             f
    13 |      260 |      3 |  0.000000E+00 |             f
    14 |      280 |      3 |  0.000000E+00 |             f
    15 |      300 |      3 |  0.000000E+00 |             f
    16 |      320 |      3 |  0.000000E+00 |             f
    17 |      340 |      3 |  0.000000E+00 |             f
    18 |      360 |      3 |  0.000000E+00 |             f
```

```
    19 |      380 |      3 |  0.000000E+00 |             f
    20 |      400 |      3 |  0.000000E+00 |             f
```

Visualização

Plot das soluções não dominadas:

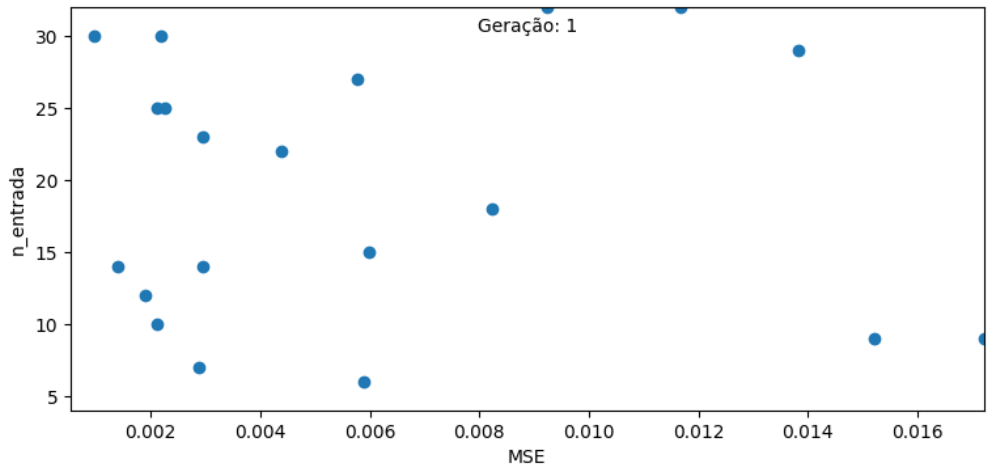
```
f1 = res.F[:,0]
f2 = res.F[:,1]
plt.scatter(f1,f2, s=30, facecolors='none', edgecolors='red',
label=('Soluções não dominadas'))
plt.xlabel("MSE")
plt.ylabel("n_entrada")
plt.legend()
plt.show()
```



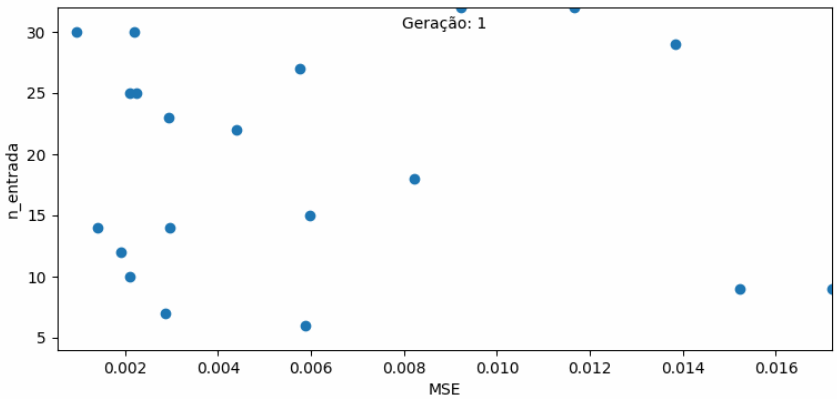
Plot dos resultados ao longo das gerações:

```
all_pop = Population()

for algorithm in res.history:
 all_pop = Population.merge(all_pop, algorithm.off)
df_Var = pd.DataFrame(all_pop.get("X"), columns=[f"X{i+1}" for i in
range(problem.n_var)])
df_Res = pd.DataFrame(all_pop.get("F"), columns=[f"F{i+1}" for i in
range(problem.n_obj)])

import matplotlib.animation as animation
from IPython.display import Image
```

```
df_Res['Modulo_F2'] = np.sqrt(df_Res['F2'] ** 2)

fig, ax = plt.subplots()
scatter = ax.scatter([], [], animated=True)
ax.set_xlim(df_Res['F1'].min(), df_Res['F1'].max())
ax.set_ylim(df_Res['Modulo_F2'].min(), df_Res['Modulo_F2'].max())
ax.set_xlabel('MSE')
ax.set_ylabel('n_entrada')

frame_counter = ax.text(0.5, 0.95, '', ha='center', va='center',
transform=ax.transAxes)

def update(frame):
    start_index = frame * ps
    end_index = start_index + ps
    data = df_Res.iloc[start_index:end_index]
    scatter.set_offsets(data[['F1', 'Modulo_F2']])

    frame_counter.set_text(f'Geração: {frame + 1}')

    return scatter, frame_counter

total_frames = len(df_Res) // ps
ani = animation.FuncAnimation(fig, update, frames=total_frames,
interval=600, blit=True)

plt.show()
```



```
writer = animation.PillowWriter(fps=1, bitrate=1800)
ani.save('scatter.gif', writer=writer)
Image(open('scatter.gif','rb').read())
```



Visualização das melhores topologias e geração das imagens dos modelos:

```
for i in range(len(best_top)):

    name = 'Modelo_' + str(i+1)
    print(name)
    print('Topologia: ')

    n_entrada = int(best_top[i][2])
    n_cnn = int(best_top[i][0])
    n_lstm = int(best_top[i][1])

    tmodel = keras.models.Sequential()
    tmodel.add(keras.layers.Reshape((n_entrada, 1),
input_shape=(n_entrada,)))

    #1st layer - CNN
    tmodel.add(keras.layers.Conv1D(n_cnn, 3, activation='relu'))
    tmodel.add(keras.layers.MaxPooling1D(2))

    #2nd layer - LSTM
    tmodel.add(keras.layers.Flatten())
    tmodel.add(keras.layers.Reshape((int((n_entrada-2)/2), n_cnn)))
    tmodel.add(keras.layers.LSTM(n_lstm, activation='tanh'))

    #Output layer
    tmodel.add(keras.layers.Dense(1, activation='sigmoid'))
```

```
    print('N. neurônios da camada de entrada: ', n_entrada)
    print('N. neurônios da camada CNN: ', n_cnn)
    print('N. neurônios da camada LSTM: ', n_lstm)
    print('MSE no conjunto de teste: ', best_sol[i][0])

    plot_model(tmodel, to_file= str(name + '_plot.png'),
show_shapes=True, show_layer_names=True)

    print('\n\n')


Modelo_1
Topologia:
N. neurônios da camada de entrada:  30
N. neurônios da camada CNN:  17
N. neurônios da camada LSTM:  32
MSE no conjunto de teste:  0.0005357632180675864



Modelo_2
Topologia:
N. neurônios da camada de entrada:  4
N. neurônios da camada CNN:  14
N. neurônios da camada LSTM:  26
MSE no conjunto de teste:  0.004422813653945923



Modelo_3
Topologia:
N. neurônios da camada de entrada:  5
N. neurônios da camada CNN:  31
N. neurônios da camada LSTM:  31
MSE no conjunto de teste:  0.0007514497265219688



print("Tempo de execução total: %s segundos " % (time.time() -
start_time))

Tempo de execução total: 3596.313062429428 segundos
```