APLICAÇÃO DE REDES NEURAIS MULTIFACETADAS NO PROBLEMA DE INFERÊNCIA DE TEMPERATURA NA USINA DE ANGRA 1

Eduardo Archanjo Cavalcante

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Nuclear, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Nuclear.

Orientador: Roberto Schirru

Rio de Janeiro
Março de 2025

APLICAÇÃO DE REDES NEURAIS MULTIFACETADAS NO PROBLEMA DE INFERÊNCIA DE TEMPERATURA NA USINA DE ANGRA 1

Eduardo Archanjo Cavalcante

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA NUCLEAR.

Orientador: Roberto Schirru

Aprovada por: Prof. Roberto Schirru
Prof. Alan Miranda Monteiro de Lima
Prof. Claudio Henrique dos Santos Grecco

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2025

Cavalcante, Eduardo Archanjo

Aplicação de Redes Neurais Multifacetadas no Problema de Inferência de Temperatura na Usina de Angra 1 /Eduardo Archanjo Cavalcante. – Rio de Janeiro: UFRJ/COPPE, 2025.

XII, 56 p.: il.; 29, 7cm.

Orientador: Roberto Schirru

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Nuclear, 2025.

Referências Bibliográficas: p. 53 – 56.

1. Redes Neurais. 2. Multiheaded Neural Network. 3. Inferência de Temperatura. I. Schirru, Roberto. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Nuclear. III. Título.

A minha família e amigos.

Agradecimentos

Em primeiro lugar agradeço ao Criador, que na sua criação eu me encanto e me motiva todos os dias a querer estar mais perto da sua presença por meio dos estudos.

Agradeço ao Professor Roberto Schirru pela orientação, discussões filosóficas e acadêmicas, que me incentivaram durante todo o mestrado.

Agradeço sempre a minha família em especial a minha mãe Giselli e Sônia minha avó que estiveram a todo momento do meu lado e despertaram em mim o interesse pela ciência, em especial a experimentação e engenharia.

Ao Marcelo, Bernardo e Victor pela parceria no laboratório que permitiram o desenvolvimento de novas ideias e que auxiliaram imensamente neste trabalho.

A todos os integrantes do LMP, que sempre se dispuseram a ajudar no que fosse preciso; e ao bom humor da equipe.

Ao Castelo. Incrível grupo de amigos (Leon, Daniel, Renan, Miguel, Carlos, Victor e Angelo) que a faculdade me deu e que certamente levarei e cuidarei por toda vida.

Aos meus amigos de profissão Vinícius e Peter pelo apoio e discussões durante esse período.

Ao meu amigo e professor Thiago Higino (*in memoriam*) pelo apoio e a introdução a física experimental e preparação para ao vestibular.

A toda equipe de suporte e apoio administrativo do PEN/COPPE UFRJ. Em especial a secretária Jô pelo apoio aos processos burocráticos.

A todos da banca examinadora, por terem aceito o convite para avaliar e contribuir nesse trabalho.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

APLICAÇÃO DE REDES NEURAIS MULTIFACETADAS NO PROBLEMA DE INFERÊNCIA DE TEMPERATURA NA USINA DE ANGRA 1

Eduardo Archanjo Cavalcante

Março/2025

Orientador: Roberto Schirru

Programa: Engenharia Nuclear

Este trabalho apresenta uma nova abordagem ao uso de redes neurais para reconstrução de dados como solução para a ausência do registro histórico de temperaturas na usina nuclear Angra 1, uma necessidade para viabilizar o pedido de extensão de sua vida útil. As condições ambientais como temperatura e radiação impactam diretamente na vida útil dos sensores ao redor do prédio de contenção, causando um envelhecimento precoce desses equipamentos diferente do que foi inicialmente calculado.

Portanto, o estudo de condições e o acompanhamento desse processo de envelhecimento são fundamentais para a criação de um panorama geral sobre a vida desses dispositivos, e é essencial para a elaboração do programa de qualificação ambiental responsável pela execução da extensão de vida útil da usina. Entretanto, a ausência de dados de temperatura em pontos estratégicos ao redor do prédio de contenção prejudica a construção da informação sobre o envelhecimento desses equipamentos.

Este trabalho amplia os estudos iniciais no uso de redes neurais artificiais para a simulação desses equipamentos, com o objetivo de inferir os dados de temperatura passadas a partir dos dados atuais. A principal contribuição desse trabalho é a apresentação da nova abordagem para o uso de redes neurais, analisando o desempenho de um modelo não sequencial e de múltiplas camadas de entradas dedicadas a cada variável do problema. Neste trabalho foi proposto um modelo que combina as arquiteturas CNN e LSTM, com múltiplas camadas de entradas responsáveis por diferentes variáveis de entrada obtendo resultados com menos parâmetros em torno de 10% sobre as redes individuais. Além disso, um teste de reconstrução simultânea para 2 variáveis obteve um resultado 60,22% melhor com relação as redes individuais e o próprio modelo em reconstrução individual.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

APPLICATION OF MULTIHEADED NEURAL NETWORKS TO THE TEMPERATURE INFERENCE PROBLEM AT THE ANGRA 1 POWER PLANT

Eduardo Archanjo Cavalcante

March/2025

Advisor: Roberto Schirru

Department: Nuclear Engineering

This work introduces a novel approach to leveraging neural networks for data reconstruction as a solution to the lack of historical temperature records at the Angra 1 nuclear power plant, a critical requirement for extending its operational lifespan. Environmental conditions, such as temperature and radiation, directly affect the lifespan of sensors around the reactor containment building, leading to premature aging of these devices, which differs from initial projections.

Monitoring these conditions and tracking the aging process is essential to providing a comprehensive overview of the lifespan of these devices. This process is crucial for developing the environmental qualification program required to support the plant's operational lifespan extension. However, the absence of temperature data at strategic points around the containment building hinders the ability to assess the aging process of these devices accurately.This work expands on initial studies in the use of artificial neural networks for the simulation of these devices, aiming to infer historical temperature data from current measurements. The main contribution of this work is the introduction of a new approach to the use of neural networks, analyzing the performance of a non-sequential model with multiple input layers dedicated to each variable of the problem. In this work, a model combining CNN and LSTM architectures was proposed, with multiple input layers responsible for different input variables. The results were satisfactory, achieving a reduction of approximately 10% in the number of parameters compared to individual networks.Moreover, a simultaneous reconstruction test for 2 variables achieved a 60.22% better result compared to individual networks and the model itself in individual reconstruction.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

AdaGrad Adaptive Gradient Algorithm (Algoritmo de Gradiente Adaptativo). 17, 18

Adam Adaptive Moment Estimation (Estimativa de Momento Adaptativo). 18

CNN Convolutional Neural Network (Rede Neural Convolucional). 3–5, 22–24, 26, 27, 29, 39, 41–46, 51

DNN Deep Neural Network (Rede Neural Profunda). 5, 22, 25, 27, 29

LMP Laboratório de Monitoração de Multiprocessos. 39

LSTM Long Short-Term Memory (Memória de Curto Longo Prazo). 3–5, 30, 31, 37, 39, 41–46, 51

LSTM-AE LSTM Autoencoder (Autoencoder baseado em LSTM). 5

LSTM-FCN Fully Convolutional Network (LSTM com Rede Totalmente Convolucional). 4, 5

MAE Mean Absolute Error (Erro Médio Absoluto). xi, 39, 42–46, 48, 49

MLP Multilayer Perceptron (Perceptron Multicamadas). 11, 22, 25

PIT Problema de Inferência de Temperatura. 5, 6, 35

PQAEE Programa de Qualificação Ambiental de Equipamentos Elétricos. 2, 33, 34

PWR Pressure Water Reactor (Reator de Água Pressurizada). 3, 5

RNN Recurrent Neural Network (Rede Neural Recorrente). 27, 29

SGD Stochastic Gradient Descent (Gradiente Descendente Estocástico). 16

SICA Sistema Integrado de Computadores de Angra. 2

Capítulo 1

Introdução

É amplamente conhecido que usinas nucleares são sistemas complexos que são projetadas e operadas sob diretrizes de segurança rigorosas, que devido a possibilidade de ocorrência de acidentes que levam a consequências catastróficas tem como prioridade a segurança em primeiro lugar, de forma a manter os riscos de acidentes tão baixos quanto razoável para o funcionamento. Contudo, apesar dos riscos envolvidos, a energia nuclear se mostra como uma das principais soluções para a geração de energia segura e limpa para o combate a produção de gases do efeito estufa, além de contribuir com a prevenção de milhões de mortes decorrentes da emissão desses gases [1].

Além disso, com o aumento da temperatura média global e o agravamento das crises climáticas, fontes de energia renováveis intermitentes que dependem de condições favoráveis do clima, como usinas hidrelétricas são severamente prejudicadas com a frequência da ausência de chuvas, principalmente em solo brasileiro, onde essa fonte corresponde a mais 61% da produção total de energia [2]. Com isso, a longo prazo, caso a situação global não melhore, outra forma de produção de energia limpa intermitente e de alta produtividade deve ser ampliada a fim de evitar a escassez por falta de reservatórios hídricos.

Neste cenário se destaca a energia nuclear como alternativa de produção limpa de energia e independente das condições climáticas; sua alta densidade de energia por célula de combustível e funcionamento ininterrupto colocam as usinas nucleares como fontes de energia do futuro, acompanhando as decisões ambientais e a alta demanda de energia[2]. Entretanto, a construção de novas usinas nucleares possui um custo elevado e, como já citado, rigorosas diretrizes de construção e operação; por esse motivo, diversos países estão optando pela extensão de vida útil de suas usinas.

Ao se aproximar do fim da vida útil as usinas nucleares passam a dedicar parte de seu objetivo ao descomissionamento da planta, entretanto foi observado como alternativa viável a extensão do seu período de operação, que através de estudos de

viabilidade técnica, se mostrou algo possível, e atualmente amplamente difundido e executado em todo o mundo [3]. Apenas nos Estados Unidos, no ano de 2014, 74 das 100 usinas nucleares em operação já tinham recebido licenciamento para operar além de sua vida útil inicialmente projetada, enquanto que na Europa 66 de 151 usinas já operam a mais 30 anos [3].

Embora uma usina nuclear não tenha um tempo de vida útil rigidamente definido, seus componentes possuem limitações individuais. Para estender a operação da usina como um todo, é necessário realizar uma série de estudos dentro do Programa de Qualificação Ambiental de Equipamentos Elétricos (PQAEE). Esse programa avalia se os equipamentos, sistemas e componentes essenciais para a segurança da planta continuam em boas condições de funcionamento, permitindo sua permanência na usina durante o período de extensão da vida útil. [4].

Neste contexto, a usina nuclear de Angra 1 encontra-se, na data do presente trabalho, ao fim de sua vida útil padrão de 40 anos, e no contexto nacional da produção energética as usinas de Angra suprem cerca de 40% da energia do estado do Rio de Janeiro e 3% da matriz nacional, colocando a usina em um cenário especial, onde impacto de seu desligamento seria considerável. Portanto, dada a importância da usina, está em andamento o processo de extensão de sua vida útil por mais 20 anos[5]. Entretanto, por obter seu licenciamento de construção antes da elaboração das normas que estabelecem o PQAEE para usinas nucleares, a usina de Angra não possuiu um plano de qualificação ambiental estabelecido desde sua construção[4].

Dessa forma, a fim de mapear as condições ambientais no qual operam os equipamentos elétricos que compõem o PQAEE em condições normais de operação, foram instalados no interior do prédio de contenção da usina conjuntos de monitoração de parâmetros de dose de radiação e temperatura (SMOs). Essa monitoração será feita até próximo ao fim de vida útil da instalação, a fim de obter um panorama das temperaturas e doses de radiação nas áreas onde os equipamentos estabelecidos no PQAEE operam, e seja possível dizer as condições de operação de seu funcionamento ao longo desses anos; e por fim, calcular a vida qualificada desses equipamentos e saber quais devem ser substituídos, recondicionados ou continuar funcionando[4]. Entretanto, a instalação desses equipamentos começou 2015, logo, não há dados que mapeiem a temperatura desses ambientes antes desse período, dificultando a elaboração do estudo de envelhecimento desses equipamentos com relação a temperatura desde o início da usina, limitando apenas a partir desse ano.

Contudo, Pinheiro [4] demonstra em sua tese a utilização de redes neurais artificiais com base nos dados do Sistema Integrado de Computadores de Angra (SICA), que monitora diversas variáveis físicas de operação da usina, que estão diretamente correlacionadas a temperatura no interior do prédio de contenção, para a inferência dos dados de temperatura desse período sem monitoração.

O objetivo deste trabalho é o estudo da aplicação de redes neurais multifacetadas, ou de arquitetura arbitrária, no Problema de Inferência de Temperatura na usina de Angra 1. Este estudo visa complementar os trabalhos feitos por Pinheiro [6] e Cardozo [3] com a apresentação de um novo modelo de arquitetura de rede neural, utilizando uma abordagem de rede não sequencial e múltiplas camadas de entradas dedicadas[7] que combina diferentes arquiteturas Convolutional Neural Network (Rede Neural Convolucional) (CNN) e Long Short-Term Memory (Memória de Curto Longo Prazo) (LSTM) em um só modelo, de forma a extrair suas melhores características no que diz respeito a extrair características e correlacionar séries temporais, respectivamente, para obter melhores resultados na inferência de temperaturas.

Com resultados promissores, esse trabalho inicia o estudo da viabilidade dessas arquiteturas para problemas de inferência de dados, se beneficiando da modularidade de flexibilidade desse modelo de forma a criar uma base de estudo para implementação de um eventual modelo de rede único multipropósito, ou seja, capaz de reconstruir dados para diferentes sensores simultaneamente, e ainda permitir a adição de novas camadas dedicadas a novos equipamentos.

Como referência para esse trabalho foram consideradas as principais obras literárias usadas no estudo de redes neurais e outras publicações no âmbito da engenharia nuclear que utilizaram diferentes arquiteturas de redes, especialmente em trabalhos na área de engenharia de fatores humanos, que visam otimizar a operação e manutenção de usinas nucleares do tipo Pressure Water Reactor (Reator de Água Pressurizada) (PWR). Além disso, outros trabalhos considerados relevantes no que diz respeito a novas aplicações e otimização de redes neurais são usados como embasamento teórico para esse trabalho. Portanto, a seguir, será apresentado uma breve descrição do avanço das redes neurais e das arquiteturas relevantes para esse trabalho, além dos trabalhos relevantes da engenharia nuclear que destacam seu uso em engenharia de fatores humanos com foco em Usinas nucleares do tipo PWR.

Em 1943 MacCulloch e Walter Pitts [8] propuseram a primeira ideia de neurônios artificiais que operam cálculos lógicos, na tentativa de emular o funcionamento do cérebro humano através de um sistema matemático composto de células unitárias. Neste sentido, mostraram que o neurônio pode ser representado como um processador binário que recebe entradas de outros neurônios e as combina para produzir uma saída binária (0 ou 1); ainda demonstraram a capacidade que a combinacao dessas células é capaz de executar funções lógicas básicas como *and*, *or* , *not* , etc [8].

Poucos anos depois, em 1947, Donald Hebb propôs a ideia de aprendizado e treinamento dessas células por meio da modificação dos pesos que compõem as conexões entre dois neurônios através das suas interações com os dados e resultados. Tornando assim, a primeira apresentação de algoritmo de aprendizado supervisionado de redes

neurais artificiais [9].

Em 1950, Frank Rosenblatt apresentou o perceptron como primeira construção real computacional de um neurônio artificial proposto por MacCulloch representando as diversas partes do funcionamento natural de um neurônio biológico como uma função que opera uma soma ponderada e ainda o conceito de impulso elétrico que passado adiante a partir de um limiar, representado por uma função de ativação. Entretanto o modelo do perceptron se mostrou simples demais para a aplicação em problemas não lineares mais complexos, mas permitiu a visão inicial dos primeiros conceitos de uma rede neural artificial [10].

Em 1986, após um período de relativa estagnação nas pesquisas de redes neurais devido às limitações do modelo de perceptron, seja pela simplicidade de problemas aplicáveis, a problemas de treinamento para múltiplos perceptrons conectados, o interesse foi revitalizado pela introducao das redes multicamadas e do algoritmo de treinamento backpropagation apresentados por Rumelhart, D. E., Hinton, G. E., & Williams [11].

No que diz respeito ao uso de redes neurais para aplicação de processamento de dados em sequências temporais, foi proposto inicialmente o conceito de redes neurais com memória por John Hopfield em 1982 com a introdução da Rede Hopfield, sendo um dos modelos iniciais de redes neurais recorrentes, que se popularizaram na década de 1980 e 1990[12].

Em 1991, Hochreiter e Bengio (1994), identificaram o problema de desaparecimento de gradiente que impossibilitava os modelos tradicionais de redes neurais recorrentes de relacionar dados de sequências longas, ficando presas a pequenos trechos de algumas dezenas de passos temporais [13].

Em 1997, Hochreiter e Schmidhuber apresentaram a arquitetura de redes neurais recorrentes de longa memória curta (LSTM), uma rede capaz de definir quais dados passados são relevantes a serem considerados para contribuir na memória construída da rede; se tornando uma das principais arquiteturas de redes recorrentes para problemas de sequência[14].

No campo da visão computacional se destaca o trabalho de Yann LeCun 1998, onde apresenta o modelo LesNet, uma das primeiras redes convolucionais capaz de reconhecer dígitos manuscritos [15].

Em 2019 Karim introduz a arquitetura Fully Convolutional Network (LSTM com Rede Totalmente Convolucional) (LSTM-FCN) que combina o modelo LSTM com uma rede CNN, para classificações de séries temporais. Essa metodologia se beneficia das características das redes LSTM em capturar as dependências em series temporais e da aptidao das CNN em extrair as características dos dados. Karin, portanto, mostra que essa combinacao se sobressai de modelos mais atuais de redes neurais em 28 dos 35 testes realizados[16].

No que tange o uso de redes LSTM auto codificadas na área de fatores humanos aplicado a usinas nucleares do tipo PWR se destaca o trabalho de Santos, com o uso dessa arquitetura para identificar tipos de acidentes nucleares. Santos, treinou 16 redes LSTM Autoencoder (Autoencoder baseado em LSTM) (LSTM-AE) para reconstruir os dados de sequência temporal de 16 acidentes diferentes, obtendo resultado de 100% de acerto nas classificações dos acidentes treinados [17].

Em 2020 Pinheiro apresenta em sua tese a proposta de solução para o Problema de Inferência de Temperatura (PIT) para a usina de Angra 1, usando as arquiteturas LSTM e DRNN para reconstruir os dados de temperatura não observáveis em instantes de tempos onde sensores mais atuais não estavam presente, usando dados de sensores reais da usina em posições nos quais os dados sao correlacionados. Dessa forma, Pinheiro conseguiu inferir dados de temperatura passados com um erro de 2°C em 89% dos casos aplicados[4].

Em 2022 Cardozo apresenta uma comparação dos modelos tradicionais de DNN com a rede LSTM, tendo como referência os trabalhos de Pinheiro, mostrando um desempenho de 25% melhor para os modelos LSTM em comparação às arquiteturas Deep Neural Network (Rede Neural Profunda) (DNN) [3].

Em 2023 Santos apresenta a comparação de diversos modelos de arquiteturas na identificação de acidentes, em especial a combinação LSTM-FCN introduzida por Karim [6]. Marcelo demonstra a evolução do desempenho ao aumentar o tamanho da janela deslizante que o modelo observa a sequência temporal do acidente; além disso aplica a ideia apresentada por A. dos Santos [17] no desenvolvimento de um sistema baseado em uma LSTM-AE para validar os resultados não identificados corretamente como "não sei".

Neste trabalho, foi desenvolvido um modelo que combina as arquiteturas CNN e LSTM, utilizando múltiplas camadas de entrada para processar diferentes variáveis de forma simultânea. O modelo combinado apresentou desempenho superior, alcançando resultados satisfatórios com uma melhoria de aproximadamente 10% em relação às redes individuais, enquanto utiliza menos parâmetros treináveis. Além disso, um teste de reconstrução simultânea para 2 variáveis obteve um resultado 60,22% melhor com relação as redes individuais e o próprio modelo em reconstrução individual.

A organização desse trabalho é definida da seguinte maneira:

No Capítulo 1 foi feita a introdução ao problema abordado, destacando o objetivo deste trabalho junto com as referências bibliográficas usadas como base para a construção da ideia e aplicação nesta dissertação.

No Capítulo 2 é apresentado a fundamentação teórica das redes neurais, priorizando as arquiteturas e modelos utilizados neste trabalhos.

O Capítulo 3 descreve com mais detalhes as arquiteturas estudadas para cada abordagem do PIT, explicitando todos os hiperparâmetros, funções e bibliotecas utilizadas para a criação dos modelos neurais, suas motivações e a escolha dos dados usados para processamento.

O Capítulo 4 apresenta os resultados obtidos para cada abordagem e compara com trabalhos anteriores com diferentes escolhas de arquiteturas e modelos.

O Capítulo 5 apresenta as conclusões do trabalho junto com algumas sugestões para trabalhos futuros e outras possíveis aplicações das ideias apresentadas.

Capítulo 2

Fundamentação Teórica

2.1 Redes Neurais Artificiais

Redes neurais são modelos matemáticos que buscam abstrair o funcionamento do cérebro humano em um conjunto de neurônios artificiais que conectam entre si formando uma cadeia de transmissão de informações através de Equações matemáticas. Em 1943, Warren Culloch [8], visando entender esse funcionamento, formulou os primeiros conceitos de neurônios artificiais, e mais tarde, Rosenblatt [10] em 1950, como conexão entre nós como mostrado na Figura 2.1, que operam multiplicação entre matrizes de pesos e dados de entrada, simulando a passagem de impulsos elétricos:

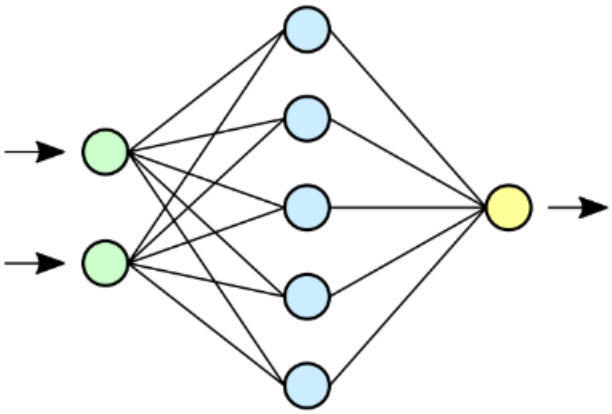

Figura 2.1: Rede neural MLP simples proposta por frank Rosenblatt
Fonte: `https://pt.wikipedia.org/wiki/Rede_neural_artificial`

Essa estrutura pode ser modelada de maneira a se adequar a diferentes tipos de problemas matemáticos, desde regressão linear a reconhecimento de padrões. Suas principais características são: a abstração do aprendizado, onde não há a dependência de programar instruções claras para resolver o problema, visto que sua

principal função é buscar uma relação entre os dados de entrada e saída da função que se busca aproximar sem se importar fluxo de código; e velocidade de execução após o processo de treinamento ser concluído, visto que, matematicamente, o conhecimento das redes neurais representado por pesos sinápticos são matrizes que operam os dados de entrada transformando em saída, sendo computacionalmente uma tarefa trivial para computadores modernos [9].

2.1.1 Neurônio Biológico

O neurônio é a unidade básica da estrutura cerebral. É a célula responsável pelo processamento unitário de informação através do cérebro. A combinação de diversos neurônios é chamada de Rede Neural, uma rede complexa de bilhões de células interconectadas, no qual a informação é transmitida por impulsos elétricos através das conexões intercelulares. O neurônio é constituído de três partes, sendo a primeira o corpo, onde a informação é processada (por meio de processos químicos e elétricos), a segunda os dendritos, que são prolongamentos por onde o corpo da célula recebe a informação de outras células, e por fim o axônio, uma extensão maior do corpo onde a informação recebida pelos dendritos e processada pelo núcleo é transmitida a outros neurônios como demonstrado na Figura 2.2.

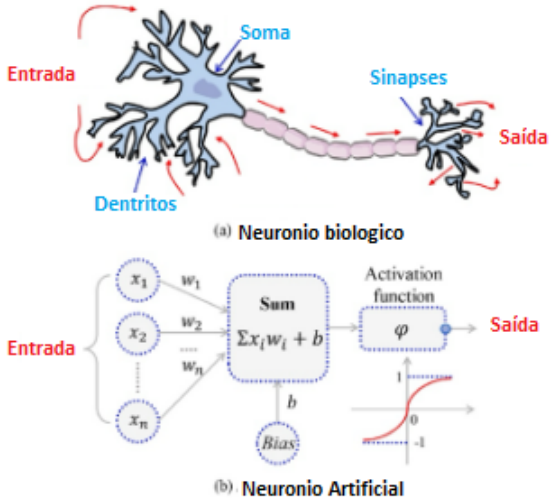

Figura 2.2: Representação do neurônio biológico [18]

O processo de tratamento e transmissão de informação em um neurônio é feito através de estímulo e resposta. O impulso elétrico recebido pelos dendritos chega ao corpo, onde um estímulo é captado e, caso este ultrapasse um limite de potencial elétrico (threshold), o sinal é passado adiante através do axônio a outro neurônio onde o mesmo processo será executado. Esse processamento com base em *threshold*

explica como um conjunto de neurônios interconectados se adapta a diferentes tipos de estímulos provenientes de nervos conectados em todo o corpo humano, formando uma cadeia de conhecimento e memória.

Com base nesse conhecimento do funcionamento do sistema neuronal, Warren McCulloch e Walter Pitts [8] propuseram a criação do neurônio artificial matemático, buscando recriar a maneira como o cérebro retém informações e aprendizado em um sistema eletrônico puramente artificial.

2.1.2 Neurônio Artificial

Semelhantes aos neurônios biológicos, os artificiais funcionam como uma unidade de processamento, onde os impulsos (*inputs*) vindo dos dendritos são processados no corpo e, caso o valor final da operação esteja dentro de um intervalo definido, é transmitido o sinal para outro neurônio. Por analogia, os dendritos em um neurônio artificial são os pesos W que multiplicam os valores do vetor de entrada X recebido da conexão de outras células. De mesmo modo, o processamento no corpo consiste em um somatório das entradas multiplicadas pelos seus respectivos pesos, e somado ao valor final um bias B, como apresentado na Figura 2.3.

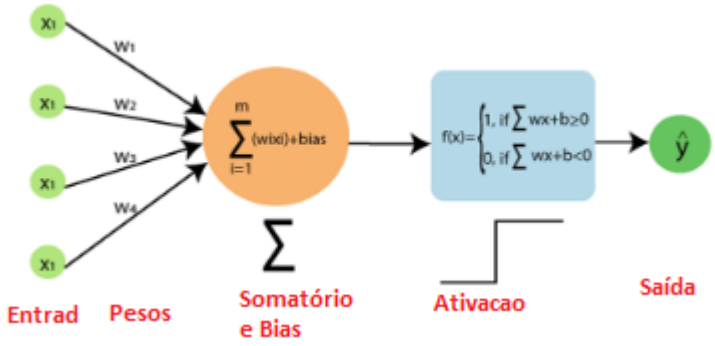

Figura 2.3: Representação do neurônio artificial [19]

Ao final da operação o valor é passado como parâmetro para uma função ativação que funcionará como checagem de *threshold* e definirá se o sinal será passado adiante como um valor de representação positiva (ligado) ou negativa (desligado) a outro neurônio. Dessa forma, com essa analogia matemática, o neurônio artificial é tratado como uma unidade computacional que opera cálculos numéricos com vetores de entrada e saída (*output*), sendo representado matematicamente como um vetor de pesos w que multiplica os vetores de entrada x.

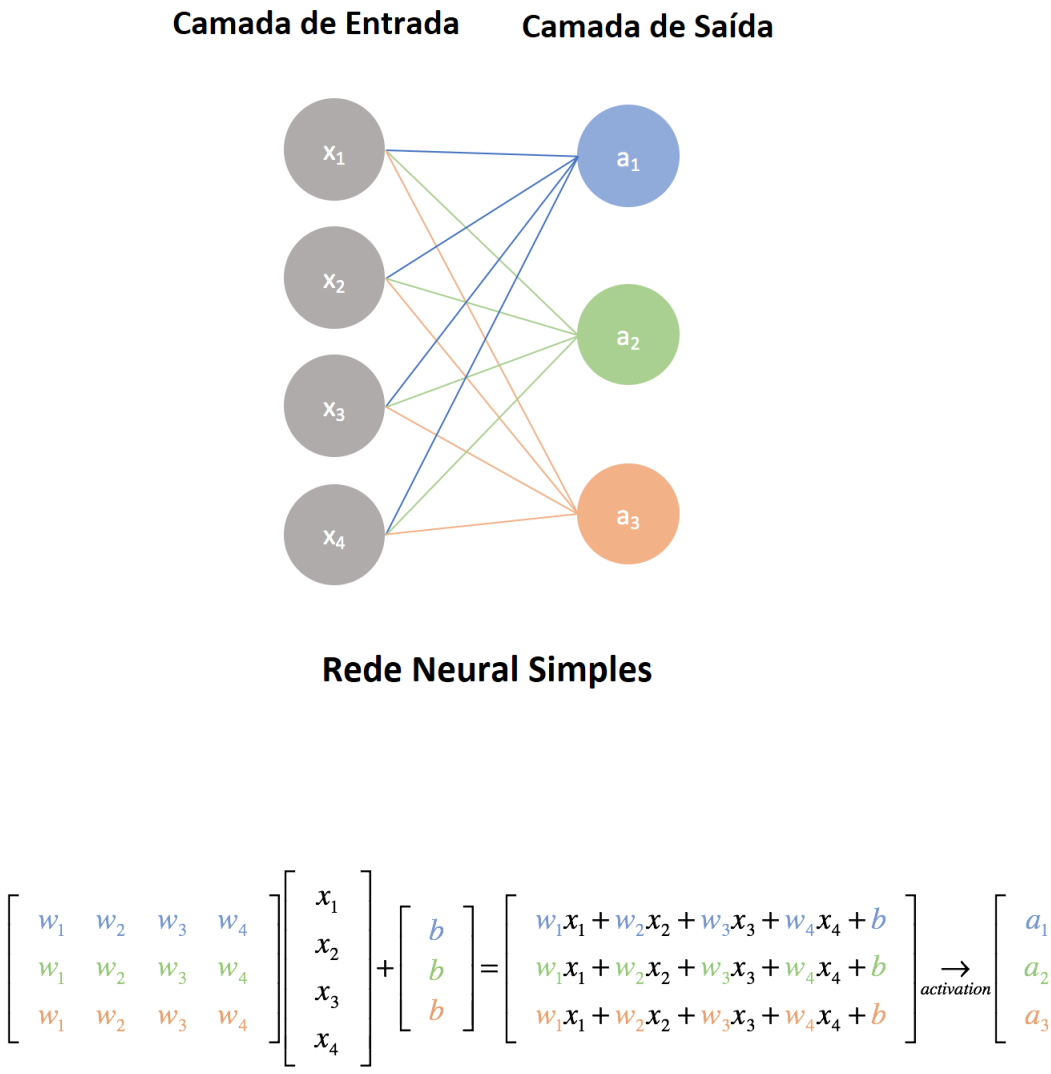

Figura 2.4: Representação matricial do neurônio artificial
Fonte: `https://www.jeremyjordan.me/intro-to-neural-networks/`

Além disso, no âmbito da computação, o neurônio artificial é chamado de Perceptron [10], e sua concepção em 1958 por Frank Rosenblatt o define como a unidade mais básica de uma rede neural e consiste em um único neurônio matemático com uma função de ativação $f(x)$ definida pela Equação 2.1. Sua principal característica é a capacidade de resolver problemas linearmente separáveis, onde há apenas duas possíveis respostas (comumente usadas como 0 ou 1).

$$f(x) = \begin{cases} 1, & \text{se } w \cdot x + b \geq 0, \\ 0, & \text{caso contrário} \end{cases} \tag{2.1}$$

Entretanto, logo notou-se a limitação dessa célula para problemas que envolvessem outras saídas possíveis (problemas não-lineares). Nesse sentido, propôs-se a combinação de perceptrons para formar diversas arquiteturas de redes neurais, no qual a interconexão de unidades de processamento simples é capaz de resolver problemas mais complexos como classificação, previsão e clusterização.

2.1.3 Redes Neurais MLP

A Multilayer Perceptron (Perceptron Multicamadas) (MLP) foi proposta por Rumelhart, D. E., Hinton, G. E., Williams em 1986 [11], introduzindo a ideia de um modelo com duas ou mais camadas de perceptrons interconectados vide a Figura 2.5, capaz de resolver problemas complexos que não fossem linearmente separáveis. As MLP, também conhecidas como redes neurais profundas do tipo *Deep Feedforward Neural Networks*, formam a base dos modelos de redes profundas, ao estabelecer o conceito de fluxo de informação através de múltiplas camadas de neurônios.

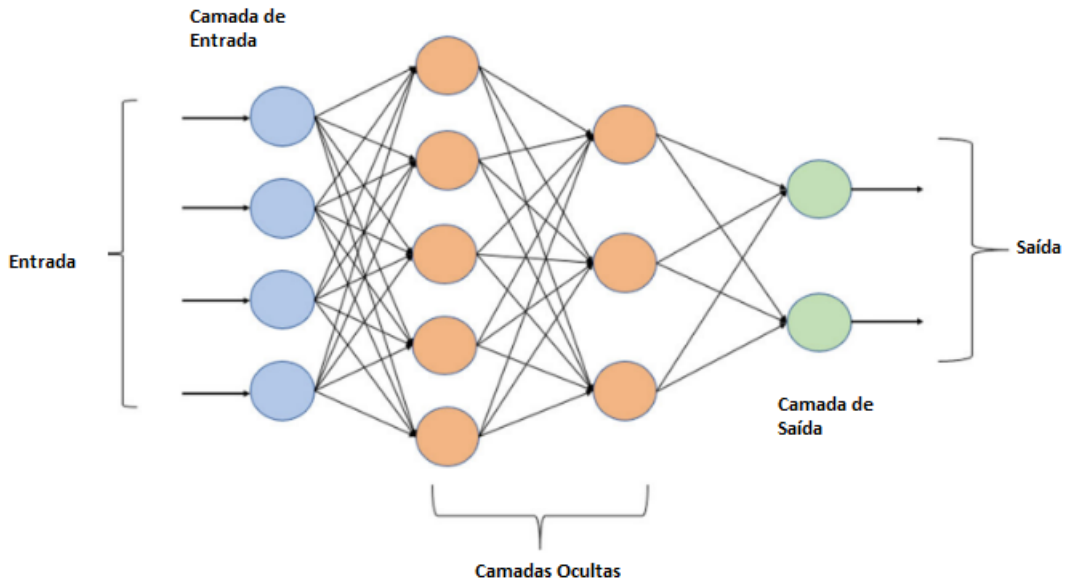

Figura 2.5: Arquitetura básica de uma rede multilayer perceptron [20]

Dado que a unidade de um perceptron pode ser representada por uma função 2.1, uma rede neural pode ser entendida como a composição de várias funções. Por exemplo, considere três funções, $f^{(1)}$, $f^{(2)}$,e $f^{(3)}$, conectadas em cadeia, formando $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$, onde $f^{(1)}$ representa a primeira camada, $f^{(2)}$ a segunda, e $f^{(3)}$ a terceira e última camada. Nesse contexto, cada função corresponde à combinação dos neurônios em uma camada, conectando-se à próxima.

Embora a MLP seja capaz de resolver problemas não lineares, na época de sua concepção não havia um método eficaz de treinamento. A complexidade decorrente

da grande quantidade de parâmetros e combinações de funções tornava impraticável o ajuste dos pesos em tempo hábil.

2.1.4 Gradiente Descendente

Antes de abordar os métodos de treinamento é preciso introduzir o conceito gradiente descendente, o que requer, inicialmente a definição de função de custo deve ser estabelecido.

A função de custo é uma medida que avalia o quão próximo o valor previsto pela rede está do valor real durante o treinamento. Ela quantifica a discrepância que o algoritmo de treinamento precisa levar em consideração ao ajustar os pesos da rede para as próximas iterações de previsão-ajuste. Com essa finalidade, a função de custo deve possuir certas características como ser diferenciável, não negativa, ter um valor mínimo igual a zero, e ser crescente para valores discrepantes do esperado. O objetivo central do treinamento de uma rede neural é minimizar essa função de custo, garantindo que a rede seja capaz de generalizar bem os resultados para todas as combinações possíveis das variáveis de entrada.

O gradiente, por sua vez, é uma ferramenta de cálculo vetorial que indica a direção e a magnitude da variação de uma função em relação a uma variável. Sua aplicação no campo das redes neurais artificiais se dá pela orientação dos algoritmos de treinamento, mostrando em que direção os pesos do modelo devem ser ajustados para reduzir a função de custo. O termo "descendente" refere-se ao movimento na direção oposta à indicada pelo gradiente, como mostrado na Figura 2.6. Sendo assim, o gradiente descendente direciona o ajuste dos pesos da rede na direção que minimiza a função de custo de forma eficiente.

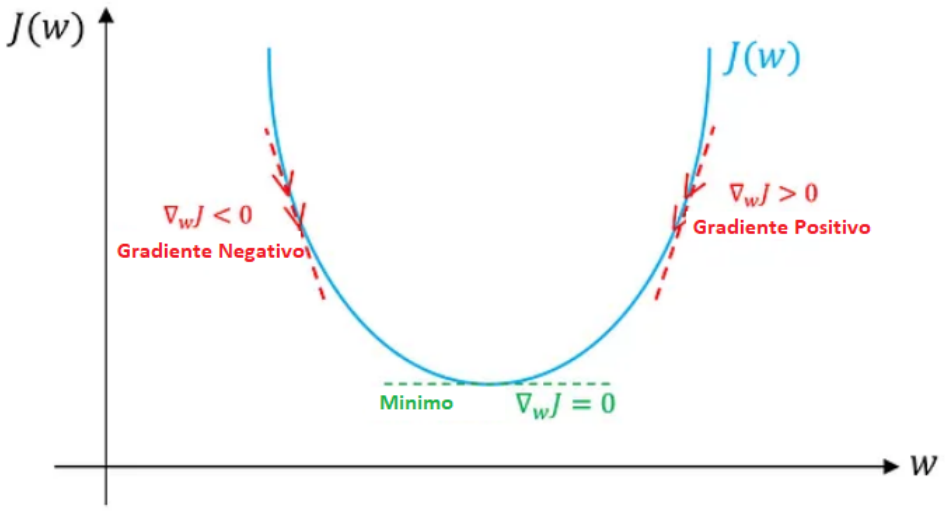

Figura 2.6: Representação do movimento do gradiente [17]

Na Figura 2.7 é possível compreender melhor a ideia de ajustar os pesos para que a função de custo seja mínima. A curva J(w) mostra a função de custo, com o valor mínimo indicado no gráfico. Ao final de cada iteração de treinamento da rede, o gradiente é calculado e indica a direção em que o algoritmo deve ajustar os pesos, usando uma taxa de aprendizado α. Com isso, espera-se que ao fim do processo de treinamento a rede encontra-se com os seus pesos ajustados de tal forma que sua função de custo está mais próxima de seu mínimo.

A taxa de aprendizado α é o valor do passo no qual o algoritmo avança em direção ao mínimo da função. É importante ressaltar a importância dos valores dessa taxa na qualidade do treinamento do modelo. Enquanto valores muito grandes podem fazer com que o algoritmo nunca alcance a região de mínimo e oscile ao redor durante todo o treinamento, valores muito baixos tornam o aprendizado lento ou coloquem o algoritmo preso em uma região de mínimo local, o que acarretaria em um aprendizado mal-sucedido.

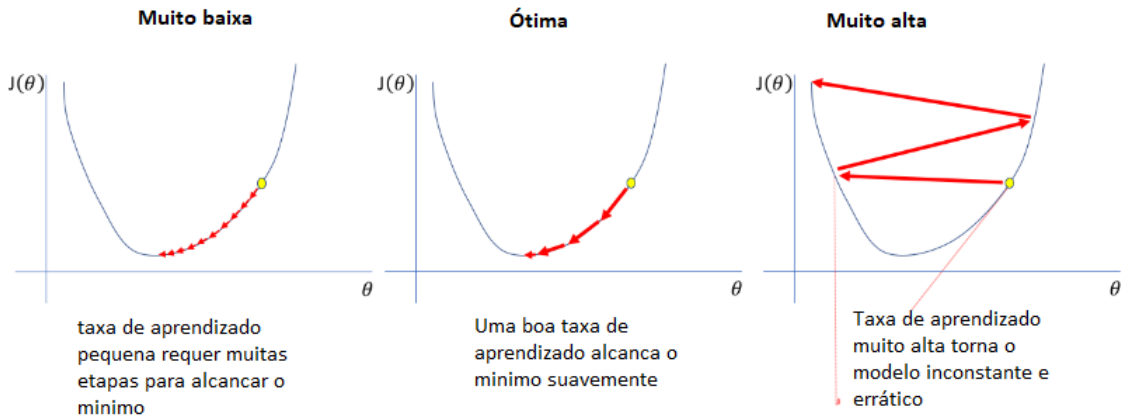

Figura 2.7: Influencia da taxa de aprendizado da rede no percurso de otimização
Fonte: `https://medium.com/grid-solutions`

2.1.5 Backpropagation

Backpropagation é um algoritmo que consiste em implementar o gradiente descendente conjunto de pesos de um perceptron multicamadas. Em resumo, calcula-se as derivadas parciais, ou seja o gradiente, de uma função aproximada $F(w, x)$ computada pela rede em relação a todos os pesos w ajustáveis, para um determinado valor de entrada x [21].

Como elaborado por Haykin [21], a técnica de *backpropagation* pode ser resumida da seguinte forma:

Seja a saída da rede neural dada pela Equação 2.2 abaixo:

$$v_j = \sum_i w_{ji} \cdot y_i + b_j \tag{2.2}$$

e

$$y_j = F(v_j) \tag{2.3}$$

onde:

- v_j é o potencial de ativação do neurônio j;
- w_{ji} são os pesos das conexões entre o neurônio i e j;
- y_i é a saída do neurônio i na camada anterior;
- b_j é o bias do neurônio j;
- $f(v_j)$ é a função de ativação aplicada ao potencial de ativação.

E o erro e_j da predição y_j em relação ao valor real d_j durante o treinamento:

$$e_j = d_j - y_j \tag{2.4}$$

Realizando a retropropagação, calculando os gradientes locais para cada vetor de pesos de cada camada, começando pela camada de saída:

$$\delta_j = e_j \cdot f'(v_j) \tag{2.5}$$

Onde $f'(v_j)$ é a derivada da função de ativação em relação à saída do neurônio. Continuando para os neurônios da camada seguinte, aplicando a regra da cadeia no sentido contrário ao fluxo de predição da rede:

$$\delta_j = f'(v_j) \cdot \sum_k \delta_k \cdot w_{kj} \tag{2.6}$$

Onde δ_k é o erro da camada seguinte e w_{kj} é o peso da conexão do neurônio j com o neurônio da camada seguinte k.

Por fim, com os erros calculados, faz-se o ajuste de cada peso individualmente:

$$w_{ji} = w_{ji} + \eta \cdot \delta_j \cdot y_j \tag{2.7}$$

onde η é a taxa de aprendizado.

O processo de cálculo retrógrado dos gradientes dos pesos da rede pode ser melhor visualizado pela Figura 2.8.

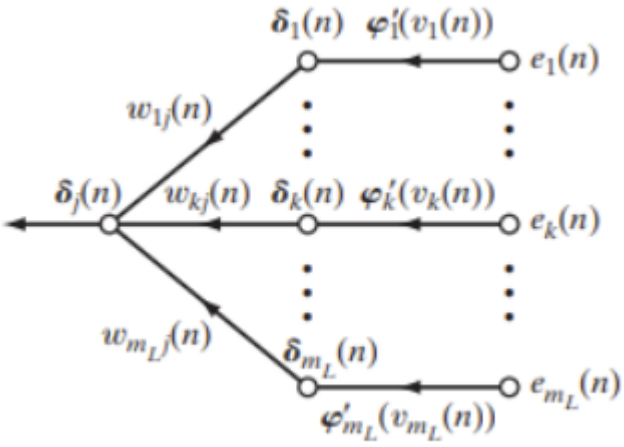

Figura 2.8: retropropagação dos erros de uma rede MLP [22]

2.1.6 Desaparecimento do Gradiente

Um dos problemas mais comuns e bem discutidos no estudo de treinamento de redes neurais é o desaparecimento do gradiente no processo de retropropagação do erro. Este incidente ocorre quando o gradiente das funções diminui a cada iteração do algoritmo, fazendo com que os ajustes dos pesos, dado pela Equação 2.7 seja insignificante, causando um aprendizado lento, ou nulo para casos onde o gradiente é zero. Este é um problema comum de redes com muitas camadas e funções de ativação no qual suas derivadas tendem a zero para valores menores, como a sigmóide[21].

Neste sentido, algumas das soluções mais usadas para contornar esse problema são:

1. Uso de funções Retificadas (RELU), que são menos suscetíveis ao desaparecimento do gradiente, pois possuem derivadas constantes para $X < 0$ e $X > 0$, 0 e 1, respectivamente;

2. Reduzir a complexidade da rede, visto que um número menor de pesos para se corrigir implica em menor possibilidade de ocorrência desses gradientes iguais a zero se propagarem.

2.1.7 Algoritmos Otimizadores

Otimizadores são algoritmos que buscam aperfeiçoar o método do gradiente descendente, reduzindo a função de custo e, em algumas técnicas, atualizando a taxa de aprendizado, de forma que o treinamento percorra maiores distâncias no espaço do gradiente da função, encontrando o mínimo global com o menor número de iterações possíveis [22].

Há inúmeros otimizadores presentes na literatura e implementados em APIs de desenvolvimento de redes neurais, e cada técnica é eficiente em tipos de trabalhos diferentes, embora seja possível utilizar qualquer otimizador. Portanto, a fim de apresentar alguns mais conhecidos e que posteriormente serão utilizados no presente trabalho; segue abaixo alguns modelos modelos:

- **SGD:** Stochastic Gradient Descent é um dos algoritmos otimizadores mais conhecidos e utilizados. Diferente do método tradicional de ajuste com base no gradiente descendente, que atualiza os pesos a cada iteração de previsão da rede no processo de treinamento, o Stochastic Gradient Descent (Gradiente Descendente Estocástico) (SGD) reduz o custo computacional dessa operação calculando o gradiente e atualizando os pesos apenas ao fim de um lote, ou "batch", de processamento da rede. O algoritmo calcula o gradiente médio desse lote m antes de prosseguir para os cálculos de ajustes [22].

- ***MOMENTUM*:** O algoritmo de *momentum* foi desenvolvido para contornar problemas de lentidão do SGD. Ele acelera o aprendizado em situações de alta curvatura do espaço de parâmetros, gradiente pequeno ou ruidosos. A ideia principal da ferramenta é a adição de um parâmetro de "velocidade" ao cálculo do gradiente, que ajuda o algoritmo a avançar o ajuste nas direções mais curvas do espaço, como visto na Figura 2.9. O termo *momentum* vem da analogia à física, no qual o gradiente negativo é a força que movimenta uma partícula no espaço de pesos. Neste sentido, a adição de momento ao cálculo do gradiente ajuda o processo de treinamento a fugir dos mínimos locais e zonas de platôs com gradiente zero [22].

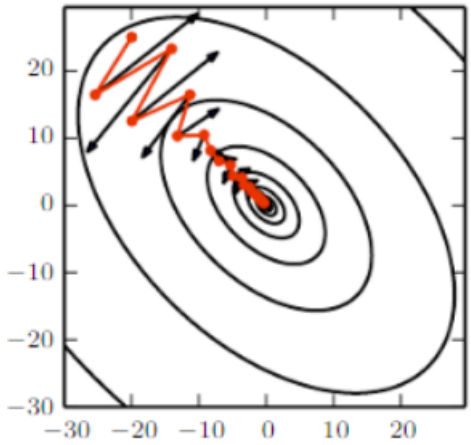

Figura 2.9: Representação gráfica do algoritmo do *momentum*
Fonte: `https://cedar.buffalo.edu/~srihari/CSE676/`

Apesar de prático, o algoritmo de *momentum* é muito suscetível ao valor da

taxa de aprendizado, visto que ao agregar momento ao cálculo do gradiente, valores altos da taxa podem fazer o algoritmo ganhar velocidade elevada e explodir o gradiente, ou mesmo ser ruidoso. Enquanto que, para valores baixos da taxa de aprendizado, o algoritmo pode entender que em certas regiões de gradiente baixo o cálculo deve ser desacelerado, tornando o treinamento mais lento.

- **ADAGRAD:** Segundo Goldfellow [22], pesquisadores de redes neurais perceberam que a taxa de aprendizado é um dos hiperparâmetros de rede mais difíceis de ajustar. Este parâmetro está diretamente ligado à sensibilidade dos ajustes dos pesos, com seus valores influenciando comportamentos dos algoritmos de otimização ou estagnando o processo de treinamento. Neste sentido, o algoritmo Adaptive Gradient Algorithm (Algoritmo de Gradiente Adaptativo) (AdaGrad) se baseia na ideia de Jacobs [23], no ajuste da taxa de aprendizado durante o processo de treinamento baseado no sinal do gradiente nas últimas iterações. Esta ferramenta ajusta individualmente a taxa de aprendizado para cada parâmetro de forma inversamente proporcional à raiz quadrada da soma de todos os valores quadrados dos gradientes anteriores g_{t+1} como mostrado na Equação 2.10. Ou seja, parâmetros que tenham os maiores gradientes, têm um decréscimo mais rápido de sua taxa de aprendizado, enquanto aqueles que possuem menores gradientes tem um leve incremento em uma taxa de aprendizado. Com isso, o algoritmo consegue percorrer com mais precisam o espaço e se ajustar melhor para regiões suavemente inclinadas [22].

$$g_0 = 0 \tag{2.8}$$

$$g_{t+1} \leftarrow g_t + \left(\nabla_\theta \mathcal{L}(\theta)\right)^2 \tag{2.9}$$

$$\theta_j \leftarrow \theta_j - \epsilon \frac{\nabla_\theta \mathcal{L}}{\sqrt{g_{t+1}} + 1 \cdot e^{-5}} \tag{2.10}$$

onde $\mathcal{L}(\theta)$ é a função de custo;

- **RMSPROP:** A fim de consertar o problema do AdaGrad onde o gradiente soma continuamente durante o treinamento, uma pequena alteração é feita ao algoritmo. Essa nova mudança, chamada de RMSPROP, adiciona um fator de decaimento , visto na Equação 2.13, na soma do gradiente, permitindo que ela diminua conforme o treinamento avança e evitando a divisão próxima de zero ao ajustar os pesos.

$$g_0 = 0, \quad \alpha \simeq 0.9 \tag{2.11}$$

$$g_{t+1} \leftarrow \alpha \cdot g_t + (1 - \alpha) \left(\nabla_\theta \mathcal{L}(\theta)\right)^2 \tag{2.12}$$

$$\theta_j \leftarrow \theta_j - \epsilon \frac{\nabla_\theta \mathcal{L}}{\sqrt{g_{t+1}} + 1 \cdot e^{-5}} \tag{2.13}$$

A ideia desse algoritmo é deixar que gradientes de regiões já muito distantes não influenciem na nova taxa de aprendizado mais recente, de forma que ela não diminua drasticamente como acontece no AdaGrad.

- **ADAM:** Proposto por Jimmy Lei Ba e Diederik P.Kingma [24], o algoritmo Adaptive Moment Estimation (Estimativa de Momento Adaptativo) (Adam) une as propriedades de *momentum* e taxa de aprendizado adaptativa dos algoritmos mencionados anteriormente. Sendo assim, ele calcula as taxas de aprendizado adaptativas individuais para diferentes parâmetros a partir de estimativas dos primeiros e segundos momentos dos gradientes; nas expressões 2.14 e 2.15, respectivamente, pode-se ver o cálculo desses momentos; além disso, é possível notar a semelhança com os algoritmos de *momentum*, AdaGrad e RMSPROP.

$$m_{t+1} \leftarrow \beta_1 m_t + (1 - \beta_1) \nabla_\theta \mathcal{L}(\theta) \tag{2.14}$$

$$v_{t+1} \leftarrow \beta_2 v_t + (1 - \beta_2) \left(\nabla_\theta \mathcal{L}(\theta)\right)^2 \tag{2.15}$$

$$\hat{m}_{t+1} \leftarrow \frac{m_{t+1}}{1 - \beta_1^t} \tag{2.16}$$

$$\hat{v}_{t+1} \leftarrow \frac{v_{t+1}}{1 - \beta_2^t} \tag{2.17}$$

$$\theta_j \leftarrow \theta_j - \frac{\epsilon\, \hat{m}_{t+1}}{\sqrt{\hat{v}_{t+1}} + 1 \cdot e^{-5} m_{t+1}} \tag{2.18}$$

Na primeira Equação, m_t é o primeiro momento dos gradientes e β_1 um fator de decaimento nesse cálculo de momento. Enquanto a segunda expressão assemelha-se ao algoritmo de RMSPROP com a adição do segundo momento e o segundo fator de decaimento β_2.

Entretanto, nota-se pela Equação (2.14) que, devido à inicialização dos momentos $m_0 = 0$ e $v_0 = 0$, o início do treinamento é severamente afetado, visto que na atualização dos pesos haverá uma divisão por um fator com valores próximos de zero, o que acarretaria atualizações grosseiras desses parâmetros logo no início do processo de treinamento. Portanto, com o intuito de corrigir esse problema, é introduzida uma correção de viés para o primeiro e segundo momentos, como visto nas Equações (2.16) e (2.17); dessa forma, as primeiras atualizações dos pesos não são grandes demais, não prejudicando o início do treinamento.

2.1.8 Overfitting e Regularização

Outro problema comum no estudo de redes neurais é o *overfitting* (ou "sobreajuste", em português). Esse fenômeno ocorre quando o modelo, durante a fase de treinamento, se especializa excessivamente nos dados utilizados para o treino, perdendo a capacidade de generalizar para novos dados fora desse conjunto. Em outras palavras, a rede aprende de forma tão detalhada os padrões dos dados de treinamento que falha ao lidar com dados desconhecidos. Neste sentido, o *overfitting* pode ser causado por diversos fatores. Um deles é a complexidade do modelo, que, devido ao número excessivo de parâmetros, acaba ajustando-se até mesmo aos ruídos presentes nos dados. Outro fator relevante é a quantidade insuficiente de dados de treinamento. Quando os dados disponíveis não são suficientes para representar adequadamente o comportamento geral das variáveis em estudo, o modelo não consegue aplicar o conhecimento adquirido a novos dados. Além disso, o tempo de treinamento também desempenha um papel importante. Se o modelo for treinado por um período excessivamente longo, ele pode se ajustar demais aos dados de treinamento, comprometendo sua capacidade de prever corretamente novos exemplos[21].

Tendo isso em vista, diversas técnicas foram construídas com o objetivo de prevenir o *overfitting*, seja melhorando e diversificando o conjunto de dados; limitando o tempo de treinamento, ou mesmo aplicando pequenas regras,chamadas de regularização, ao modelo durante o treinamento a fim de reduzir a complexidade do modelo e evitando redundâncias nas camadas da rede, que tornam o conhecimento altamente especializado.

Regularização: Dropout

Uma das técnicas mais conhecidas de regularização é o Dropout [25], que consiste no "desligamento"aleatório de alguns neurônios da rede durante o treinamento, a fim de evitar que o modelo dependa de combinações específicas de neurônios para ter um bom resultado durante o treinamento. A ideia foi apresentada por Hilton em 2012 e

popularizada em 2014 no artigo *"Dropout: A Simple Way to Prevent Neural Networks from Overfitting"* [25], onde ele apresenta a ideia de "desligamento"temporário de neurônios durante a fase de treinamento e como essa técnica ajuda evitar o *overfitting*.

Neste sentido, resumindo o conceito de dropout com base na Figura 2.10, a cada neurônio é atribuída uma probabilidade $\boldsymbol{\rho}$ de ser desconectado dos demais, sendo essa probabilidade independente. Ao construir o modelo é preciso definir a proporção de neurônios em cada camada que irá passar pelo processo de dropout, sendo os valores usualmente escolhidos entre 0.2 e 0.5

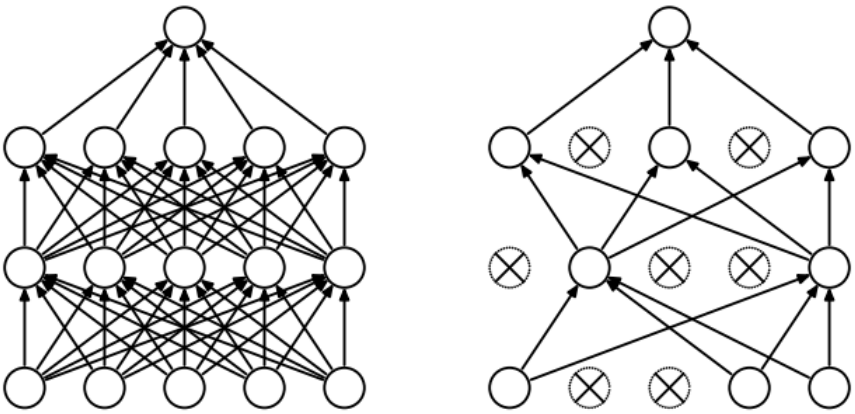

Figura 2.10: Aplicação de dropout em uma rede MLP [22]

Regularização: Decaimento de pesos

Outra técnica de regularização muito comum para evitar o *overfitting* é o decaimento de pesos [22], que adiciona à função de custo um fator de penalização, a fim de evitar o crescimento excessivo dos valores desses parâmetros, ou também promover uma esparsidade do modelo [21]. Dentro desse contexto, há dois tipos de decaimento:

- **L1 - LASSO (Least Absolute Shrinkage and Selection Operator)**: Aplica uma penalização na função de custo baseada na soma dos valores absolutos dos pesos da rede, construindo uma nova função de custo $L1$:

$$L1 = L1_0 + \lambda \sum_i |w_i| \tag{2.19}$$

  onde:

  - $L1_{\text{ori}}$ é a função de custo original;
  - λ é o fator de penalização;

– w_i são os pesos do modelo proposto.

Essa regularização penaliza valores de pesos muito elevados e força alguns a se tornarem zero, o que causa uma esparsidade no modelo, ou seja, o modelo fica menos complexo. É uma técnica útil em trabalhos que requerem uma seleção de características, onde o modelo identifica no treinamento as variáveis mais relevantes ao problema.

- **L2 - Ridge**: Adiciona à função de custo uma penalização à soma dos quadrados dos valores dos pesos, o que força os pesos a não crescerem de forma descontrolada, porém, não os deixa chegar a zero. A função de custo $L2$ fica:

$$L2 = L2_0 + \lambda \sum_i w_i^2 \tag{2.20}$$

onde:

– $L2_0$ é a função de custo original;

– λ é o fator de penalização;

– w_i são os pesos do modelo proposto.

Diferentemente da regularização $L1$, a $L2$ mantém os valores dos pesos acima de zero enquanto os impede de crescer a valores muito altos, o que permite uma maior uniformidade dos pesos na rede e reduz a necessidade de eliminar elementos da rede, permitindo um aprendizado mais suave aos dados. Neste sentido, essa técnica se aplica a problemas onde as variáveis estão altamente correlacionadas e contribuem igualmente no aprendizado [21].

EarlyStopping

No processo de treinamento, é esperado e natural que os erros de treinamento e validação diminuam com o tempo, indicando que o modelo está se aperfeiçoando, ajustando corretamente seus parâmetros e aprendendo com os dados de entrada. No entanto, é comum observar que, à medida que o tempo de treinamento se estende, o erro de validação começa a aumentar, enquanto o erro de treinamento continua a diminuir, sinalizando que o modelo está sofrendo de *overfitting*. Nesse contexto, além dos métodos de regularização já discutidos, um procedimento simples chamado Early Stopping pode ser aplicado para evitar essa divergência nos erros [22].

O Early Stopping consiste em monitorar uma métrica específica (como erro de validação ou acurácia) durante cada época do treinamento e interromper o processo caso não haja melhora em um número pré-determinado de épocas consecutivas, conhecido como "paciência" (patience, em inglês). Essa técnica, embora simples,

pode prevenir que o modelo continue treinando por épocas desnecessárias e ajuda a encerrar o treinamento antes que o algoritmo entre em uma região fora do mínimo local ou global.[21]

2.1.9 Redes Convolucionais

Com os problemas de treinamentos de modelos MLP com mais de 3 camadas solucionados pelas técnicas de *backpropagation* e o uso de algoritmos otimizadores, novas arquiteturas de redes neurais mais complexas foram desenvolvidas ao longo dos anos, e diversos problemas e desafios eram recorrentemente solucionados pelas famosas DNNs; desde problemas envolvendo classificação de múltiplas classes, otimização de processos industriais e até mesmo na área médica. Entretanto, uma área no qual esse modelo não alcançou os melhores resultados foi o processamento de dados que possuem topologia em grade. Por exemplo, series temporais com amostras em tempos regulares (grade 1D), e dados de imagem, tratadas na computação com uma matriz de pixels (2D).

Em 1998, o pesquisador francês Yen Lecun propôs o modelo de rede convolucional [26]. Um uma arquitetura de rede especializada em tratar esses tipos de dados no qual os modelos tradicionais DNNs (*Deep Neural Network*) não são eficazes. O termo convolucional se deve ao fato desses modelos usarem operações de convolução ao invés de operações matriciais entre, pelo menos, duas de suas camadas[22].

Ao comparar os modelos de Redes Neurais Convolucionais (do inglês *Convolutional Neural Network - CNN*) com as DNNs, como mostrado na Figura 2.11, é possível notar a diferença de estrutura de entrada e o núcleo dessas arquiteturas. O modelo convolucional possui uma maior flexibilidade em tratar diferentes tipos de dados, visto que a estrutura de seus neurônios em 2D pode comportar diversas entradas de dados, sejam imagens ou séries temporais. Além disso, frente as DNNs, as redes convolucionais conseguem extrair características de objetos em imagens e reconhecê-los em diferentes posições, ao contrário das redes neurais profundas que, ao possuir uma organizações de neurônios em formato vetorial ela obrigatoriamente transformaria os dados de uma imagem para um formato de uma dimensão, o que a faz perder a capacidade de relacionar as informações espaciais da imagem [17].

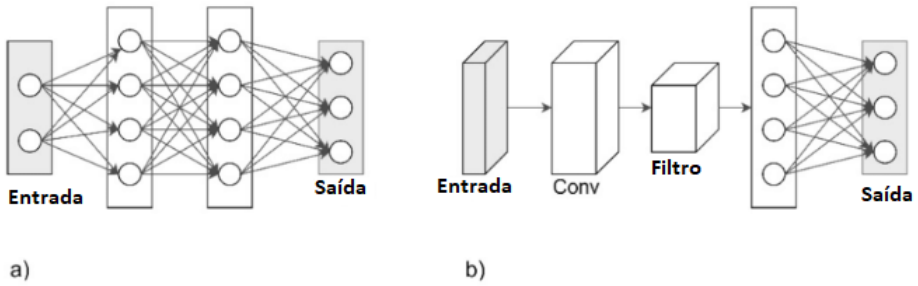

Figura 2.11: Diferença entre Arquitetura tradicional MLP e Convolucional
Fonte: `https://april21st.tistory.com/107`

Operação de convolução

A Convolução é uma operação matemática entre duas funções $f(x)$ e $g(x)$ definida pela Equação 2.21, onde o símbolo * significa a operação de convolução entre duas funções e a parte a direita, integral de convolução [17]

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(u)g(x-u)\,du \tag{2.21}$$

Como demonstrado por Santos [17], à operação de convolução acima é apenas para um espectro contínuo de tempo contínuo. Em contrapartida, nas CNNs a operação é usada no tempo discreto, onde $f(x)$ e $g(x)$ se tornam, respectivamente, os vetores discretos x, vetor de entrada e w o filtro, ou kernel da camada em questão. Além disso, um processo padding é necessário nos vetores de entrada, para garantir que as informações das bordas dos dados (sejam sequências ou imagens) sejam mantidas [22]. Esta etapa consiste na adição de zeros nas bordas da variável de entrada, sendo ela um vetor ou matriz.

Neste sentido, supõe-se x e w, respectivamente com tamanhos n e m, onde $m <= n$ e invertendo-os, tem-se a Equação de convolução discreta aplicada às CNNs:

$$x[n] * w[n] = \sum_{k=0}^{k=m-1} x^p[i+m-k]w[k] \tag{2.22}$$

Como ilustrado na Figura 2.12 abaixo, utilizando a Equação 2.22, o produto escalar $x[i; i+m].w^r$ é feito em iterações por um fator de janela deslizante (*stride*) em toda a entrada x pelo filtro rotacionado w^r [17]. Esse fator stride, um hiperparâmetro

da rede convolucional deve ser positivo e menor que o tamanho da entrada do filtro.

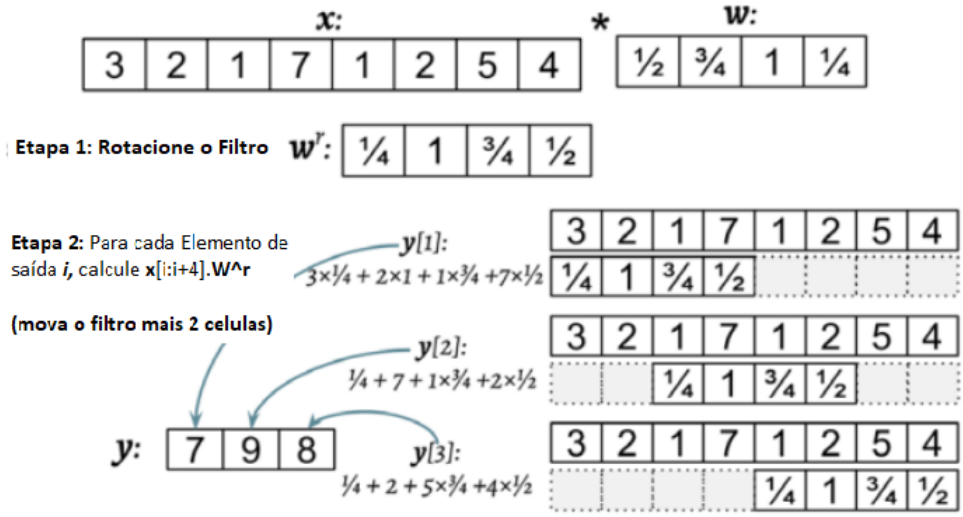

Figura 2.12: Demonstração da operação de convolução para um vetor 1D [17]

De igual modo, esta operação por ser executada com entradas em filtros de duas dimensões, comumente utilizada em redes CNNs para processamento de imagens 2D. Com isso, a Equação 2.22 aplicada para a operadores de duas dimensões fica da seguinte forma:

$$Y = X * W = Y[i,j] = \sum_{k_1=-\infty}^{+\infty} \sum_{k_2=-\infty}^{+\infty} X[i-k_1, j-k_2] W[k_1, k_2] \tag{2.23}$$

Onde $Y[i,j]$ é o pixel na posição i,já da matriz resultante da operação.

Camada de convolução

Ainda sobre a definição de uma CNN, a principal característica que define esse tipo de rede e que a difere dos modelos tradicionais de redes neurais é a estrutura da camada convolucional, comumente colocada e apresentada nas camadas ocultas dessa rede [17]. Esta camada é definida pelo agrupamento em formato de grade de neurônios, chamado de mapa de características, do inglês, *features maps*. Esse mapa é o resultado da operação de convolução da camada convolucional com a anterior e seus pesos, chamados de kernel. A ideia por trás dessa construção vem de uma analogia ao sistema visual dos mamíferos e sua capacidade de extração de detalhes e características dos objetos; semelhantemente, o mapa de característica de uma rede convolucional extrai de uma entrada, geralmente uma imagem, as porções mais significativas de informação. Por exemplo, contornos da imagem, formas geométricas,

luz e sombra, objetos rotacionados, dentre outros[22]. Na Figura 2.13 é demonstrado diferentes tipos de filtros aplicados a uma imagem e seus resultados.

Figura 2.13: Demonstração de alguns filtros (Kernel) de redes convolucionais [17]

É importante ressaltar a diferença de conexões entre os neurônios de uma camada convolucional com relação aos neurônios de redes neurais profundas do tipo MLP. Diferentemente dos neurônios de uma DNN onde cada neurônio se conecta a todos os neurônio da camada anterior formando uma rede completamente conectada; nas rede convolucionais cada neurônio que forma o mapa de características se conecta apenas a uma pequena região da camada anterior, tendo seu próprio campo receptivo[17]. Além disso, durante o treinamento, esses filtros, ou pesos, que compõem o mapa de características são ajustados seguindo o mesmo processo de treinamento anteriormente mencionado, e a rede adapta esses pesos de forma a encontrar os melhores filtros que extraiam as características relevantes dos dados de treinamento.

Camada Subamostragem

Uma adição as camadas convolucionais que são de suma importância para o diferencial dessas redes no que tange o processamento e reconhecimento de imagens e padrões de dados em duas dimensões são as camadas de *pooling*, ou camadas de

subamostragem. Essas camadas são frequentemente usadas após uma camada convolucional e são responsáveis por diversas funcionalidades. Por exemplo, a redução de dimensionalidade dos mapas de características, reduzindo a quantidade de parâmetros para processar em camadas futuras, agilizando o processo de treinamento; introduz invariância nas pequenas translações e rotações, o que permite a CNN compreender objetos em diferentes posições e rotações nas imagens e entendê-los como uma mesma classe; o realce de característica mais relevantes, que descarta informações redundantes e não relevantes para um trabalho específico e dando ênfase em aspectos mais importantes durante o treinamento; dentre outras funcionalidades[27].

Outra característica importante das camadas de *pooling* é que, apesar delas executarem a operação de convolução nos mapas de características, elas não possuem parâmetros treináveis, visto que elas trabalham apenas como transformadores das saídas das camadas convolucionais, precisando apenas que seus hiperparâmetros (tamanho do filtro e *stride*) sejam definidos na criação do modelo neural.

Dentre as camadas de pooling mais utilizadas, destaca-se o $Maxpooling$ e $AveragePooling$. Ambas consiste em uma função matemática com escopo de visualização de tamanho definido que percorre o mapa de característica em um passo (*stride*) também pré-definido extraindo o valor de acordo com a regra e resultando em um mapa de características de dimensão definida. Neste sentido, de acordo com a Figura 2.14, a camada de $Maxpooling$ extrai do espaço de visualização o maior valor dessa região, enquanto que $Averagepooling$ retorna a média dos valores. Além disso, através da imagem é possível notar a utilidade dessas duas funções: O $Maxpooling$ é útil em situações onde o objeto da imagem destaca-se do fundo por valores mais altos, o que torna o filtro muito utilizado para realçar cantos e bordas de objetos claras sobre outros mais escuros; enquanto que o $Averagepooling$ suaviza o objeto da imagem, dando mais nuances na troca de cores, o que é frequentemente usado para suavizar bordas acentuadas de objetos.

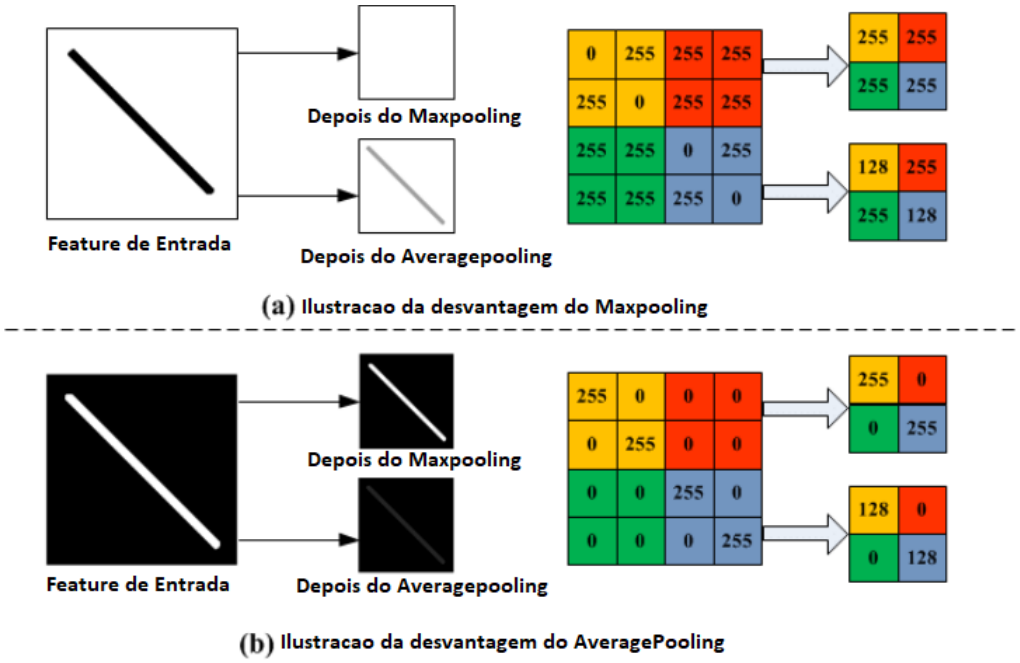

Figura 2.14: Filtros de subamostragem *Maxpooling* e *AveragePooling* [17]

2.1.10 Redes Neurais Recorrentes

Enquanto as redes convolucionais vieram como uma solução para as limitações das DNNs em processar dados em formato de grade, como imagens, diversas arquiteturas surgiram para tratar outros tipos de dados. Por exemplo, dados sequenciais, ou séries temporais. Esse tipo de dado é um problema para as arquiteturas mais clássicas justamente pelo fator sequência, que atribui aos dados uma dependência do tempo e a valores anteriores. Neste sentido, as redes tradicionais até então não possuíam formas de relacionar um valor aos seus anteriores em uma sequência, ou seja, na visão de uma DNN, por exemplo, um dado de uma série temporal era tratado apenas como um ponto no espaço relacionado ou não a uma classe ou pedido durante o treinamento.

Dado esses problemas, surgiu o modelo das Recurrent Neural Network (Rede Neural Recorrente) (RNN), uma arquitetura criada especificamente para tratar esse tipo de dados onde os valores não são independentes entre si. Além disso, semelhante às CNNs, que possuem flexibilidade para tratar imagens de tamanho diferentes, as RNNs podem ser adaptadas para diferentes tamanhos e formatos de sequências de entrada e saída, como apresentado na Figura 2.15 e detalhado abaixo [22].

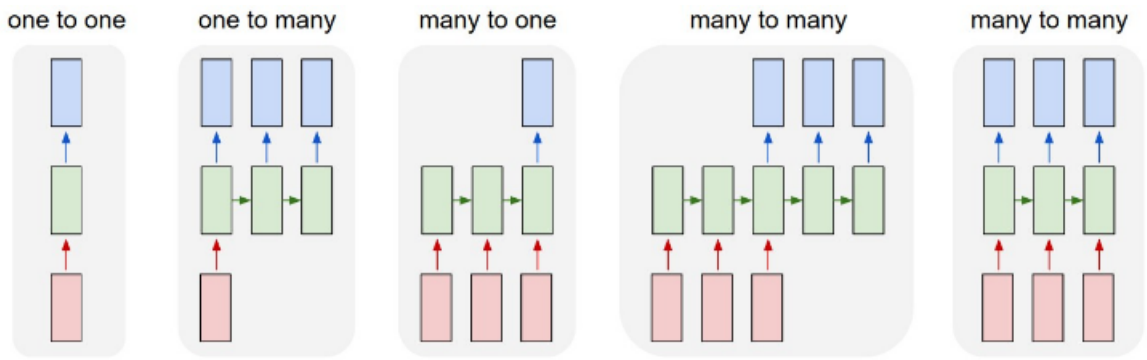

Figura 2.15: Tipos de aplicações de Redes Recorrentes [17]

- ***One to one***: Nessa estrutura tanto a entrada como a saída da rede constituem vetores de tamanho fixo. São geralmente usadas para problemas tradicionais de classificação [17].
- ***One to many***: Entrada consiste em um vetor de tamanho fixo enquanto a saída é uma sequência de vetores de tamanho dinâmico. Por exemplo, a geração de legenda ou descrição de uma imagem [17].
- ***Many to one***: Entrada é uma sequência de vetores enquanto a saída é um vetor fixo. Consiste geralmente em problemas de classificação de sequências anômalas, previsão de uma única sequência de uma variável com dados de sequências de múltiplas variáveis. Por exemplo, previsão de dados meteorológicos [17].

Os dois tipos de problemas a seguir se classificam como problemas sequência-sequência, onde ambas as entradas e saídas são sequências de vetores de tamanho variável. Além disso, pode separar esse problema em duas subcategorias de acordo com o comprimento desses vetores[17].

- ***Many to many synchronized***: O vetor de entrada e saída possuem o mesmo comprimento. Por exemplo, problemas de classificação e rotulação de um vídeo (sequência de frames)[17].
- ***Many to many Unsynchronized***: O vetor de entrada e saída possuem comprimentos diferentes. Um exemplo comum é o de problemas de tradução onde a sequência de palavras de entrada de um idioma corresponde a uma saída em outro idioma no qual a tradução se faz com menor conjunto de frases ou palavras[17].

Estrutura das Redes Neurais Recorrentes

Matematicamente, a característica principal que difere as RNNs das redes DNNs e CNNs é a aplicação de uma função de recorrência no processo de transmissão de informação nas camadas ocultas. Essa função, definida pela Equação 2.24, é chamada de relação de recorrência pois o resultado da função para um estado S_t depende do estado anterior S_{t-1} e a entrada atual da camada anterior x_t no tempo t. Dessa forma, com a informação passada recorrentemente entre os estados, a RNN é capaz de reter informações como memória de estados anteriores visto que os estados passados são relevantes para o processo de ajuste internos dos parâmetros da rede[22].

$$S_t = f(S_{t-1}, x_t) \tag{2.24}$$

Além disso, a função de recorrência não exclui o uso da função de ativação tradicional do funcionamento dos neurônios; ela é aplicada antes das funções de ativação e, em seguida, passado o estado atual para o processo de ativação do sinal que será levado adiante na rede, como mostrado nas Equações 2.25, 2.26 e 2.27, e ilustrado pela Figura 2.16[22].

Uma perspectiva visual para compreender o processo de recorrência de uma RNNs é mostrado na Figura 2.16. Este diagrama demonstra o processamento da rede recorrente de forma desdobrada (*unfolded*), ou seja, expondo as etapas de geração de sinal através dos estados em um tempo t, dependente dos estados da mesma célula em tempos anteriores. Neste sentido, ilustrado pela Figura 2.16 é matematicamente representado pelas Equações 2.25, 2.26 e 2.27, é possível entender a estrutura das redes recorrentes. Portanto, denotando as camadas de entrada, oculta e de saída para um tempo t, respectivamente por x^t, h^t e o^t, a camada de saída pode ser calculada pelas Equações 2.27[22].

$$a^{(t)} = b_1 + Wh^{(t-1)} + Ux^{(t)} \tag{2.25}$$

$$h^{(t)} = \sigma\left(a^{(t)}\right) \tag{2.26}$$

$$o^{(t)} = b_2 + Vh^{(t)} \tag{2.27}$$

Onde b_1, b_2 São os vetores de viés e U, W e V são as matrizes de pesos das conexões entrada-camada oculta, camada oculta para camada oculta e camada oculta para camada de saída, respectivamente. E a função de ativação geralmente uma função não-linear como sigmoide e tangente hiperbólica[22]. Ademais, vale dizer que, para o instante de tempo $t = 0$ as unidades ocultas recebem como estados e

entradas anteriores $t-1$ valores em zero ou próximos de zero, e apenas em estados para $t > 0$ o processo de recorrência realmente é executado[17].

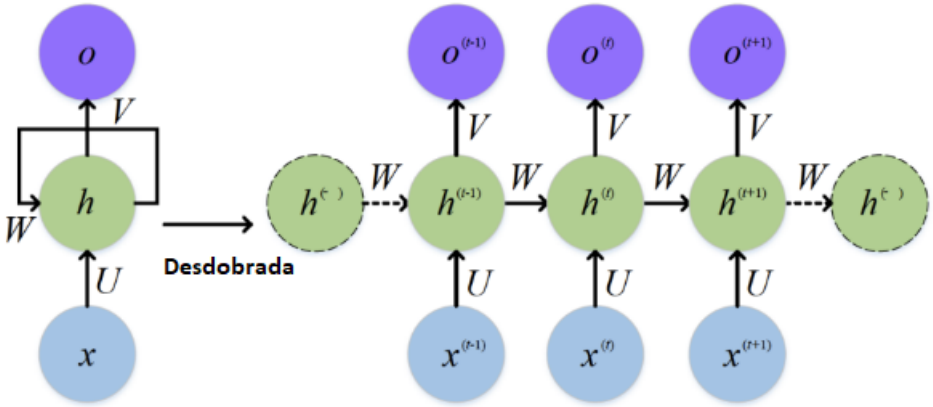

Figura 2.16: Modelo padrão de RNN e sua estrutura desdobrada(*unfolded*) [28]

Problema no treinamento de RNNs

Apesar de serem projetadas para lidar com problemas de sequência de dados, as redes neurais recorrentes também sofrem de problemas semelhantes com relação ao gradiente durante a fase de treinamento. Como mostrado por Goodfellow, as RNNs também são treinadas usando o método de otimização com base no gradiente descendente, utilizando da versão modificada chamada de retropropagação através do tempo [22]. Neste sentido, apesar da vantagem em trabalho com série temporal, ela ainda estão sujeitas ao desaparecimento do gradiente, assim como outros problemas relacionados; e como mencionado por Victor [4] e demonstrado por Hochreitter [14] os modelos tradicionais dessa arquitetura não conseguem manter a informação de dados em mais de algumas dezenas de passos anteriores, e não possuem capacidade de atribuir relevância a quais dados devem ou não ser passados adiante durante o treinamento. Sob essa perspectiva que novos modelos de redes recorrentes foram desenvolvidos para contornar essas limitações como as redes de Longa Memória Curta (do inglês *Long Short Term Memory* - LSTM)[22].

Redes LSTM

As redes LSTM apareceram em 1997, apresentada por Hochreitter & Schmidhuber [14] como um modelo que soluciona os problemas de gradiente introduzindo dentro da arquitetura as células de memória e as portas de fluxo, que controlam a passagem de informação interna da rede. Essa rede possui a mesma estrutura de entrada e saída dos modelos tradicionais de redes recorrentes, no entanto possui mais parâmetros devido ao sistema de portas. Além disso, o estado atual oculto da rede, que possui a estrutura de recorrência, é controlado pelo portão de esquecimento,

que decide o quanto da memória é adicionada ao estado atual. Com essas inclusões a rede adquire a capacidade de filtrar informações e decidir qual informação deve ser mantida e passada adiante no fluxo da rede de acordo durante o processamento. Isso garante que a rede consiga manter e gerenciar a relação temporal da informação e retenção por mais tempo[22].

Dito isso, demonstrado pela Figura 2.17, a arquitetura LSTM consiste em um bloco recorrente contendo um estado atual H_t e três portões de operação, que são: O portão de entrada I_t, que determina o quanto do valor do nó de entrada será atribuído ao estado atual da célula; o portão de esquecimento F_t, que determina se o estado anterior da memória C_{t-1} deve ser passado adiante ou descartado; e por fim, o portão de saída O_t que determina quanto do estado atual calculado pelos portões anteriores será passado adiante para os estados de estantes seguintes.

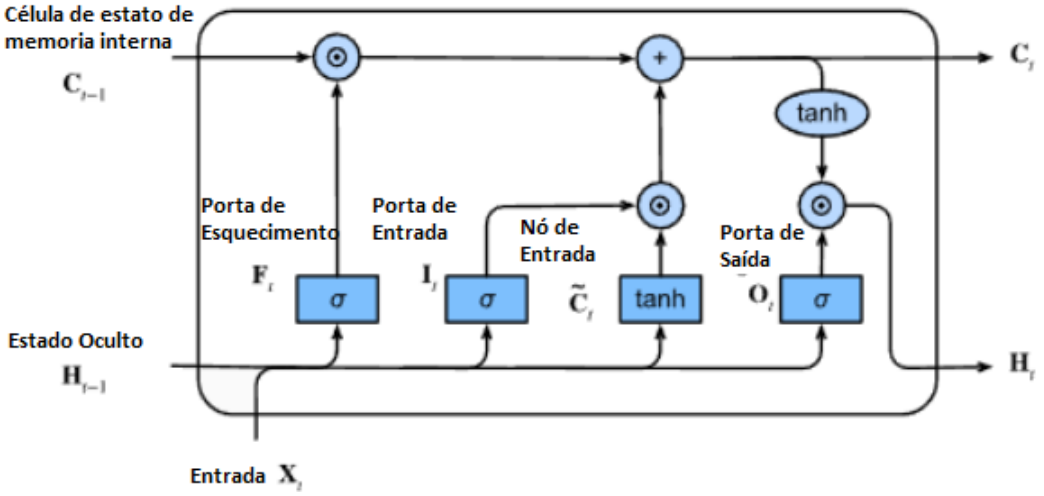

Figura 2.17: Modelo de Arquitetura LSTM [29]

Portanto, como detalhado por Kamilya Smagulova[30], as operações multiplicativas de cada portão são definidas pelas Equações 2.28,2.29,2.30,2.31:

$$g_t = \tilde{C}_t = \tanh\big(W^{(g)}x_t + U^{(g)}h_{t-1} + b^{(g)}\big) \tag{2.28}$$

$$f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}) \tag{2.29}$$

$$i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}) \tag{2.30}$$

$$o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}) \tag{2.31}$$

Por fim os estado da memória atual da célula C_t e a sua saída são calculados

pelas Equações 2.32 e 2.33:

$$C_t = g_t \odot i_t + f_t \odot C_{t-1} \tag{2.32}$$

$$h_t = o_t \odot \tanh(C_t) \tag{2.33}$$

Onde o operador $\odot$ define a operação ponto a ponto ou multiplicação de HADAMARD [30]

Capítulo 3

Introdução ao Problema e Metodologia

3.1 Problema de Inferência de Temperatura

Como apresentado introdutoriamente no Capítulo 1, para o processo de avaliação da extensão da vida útil da usina nuclear de Angra 1 ser feito é necessário a elaboração do PQAEE para identificar quais equipamentos atualmente em operação na planta estão em condições ideais de operação e seus respectivos estados de envelhecimento[4]. Além disso, para elaborar esse documento é necessário entender quais os agentes estressores, que causam o envelhecimento durante o funcionamento, que os equipamentos em operação na usina estão sujeitos.

Neste sentido, dentre os fatores externos que aceleram o processo de envelhecimento natural desses equipamentos, além da radiação ionizante e pressão, a temperatura é um dos principais causadores de degradação dos equipamentos elétricos, alterando propriedades físicas, mecânicas e elétricas responsáveis pelo pleno funcionamento. Mesmo variações de poucos graus acima da temperatura média ambiente calculada pode causar a degradação desses dispositivos ao longo do tempo através do processo chamado envelhecimento térmico. Equipamentos elétricos são especialmente suscetíveis a esse tipo de processo devido ao aquecimento natural durante sua operação, que somado a temperatura ambiente elevada agrava este processo. Por este motivo deve se levar em consideração a temperatura e seus na estimativa da qualificação ambiental desses equipamentos[4].

Sabendo-se disso, um modelo tipicamente aplicado para estimar a degradação de equipamentos em decorrência da temperatura é o modelo térmico de envelhecimento de Arrhenius que, para o contexto de avaliação da vida qualificada, é expresso pela seguinte fórmula[4]:

$$\frac{t_s}{t_a} = e^{\Phi_k}\left[\frac{1}{T_s} - 1\right] \tag{3.1}$$

- Φ: Energia de ativação (eV, sendo 1 eV = 23, 06 Kcal/mol);
- k: Constante de Boltzmann ($8,617 \times 10^{-5}$ eV/K);
- t_a: Tempo de envelhecimento acelerado (h);
- t_s: Tempo de serviço simulado (h);
- T_a: Temperatura de envelhecimento (K);
- T_s: Temperatura de serviço (K).

Este modelo estabelece que todo processo de degradação consiste em um processo químico que ocorre a partir de uma energia de ativação mínima Φ_k, que na equação 3.1 representa a energia fornecida pelo aquecimento operacional acima da temperatura de serviço T_s do equipamento, sendo esta a temperatura de de envelhecimento T_a. Portanto, este modelo de Arrhenius[31] relaciona o tempo de envelhecimento acelerado t_a, que consiste na operação do equipamento sobre uma temperatura de elevada T_a acima da temperatura de serviço previamente estipulada T_s, com o tempo de serviço estipulado t_s.

Diante disso, a fim de estender o tempo de operação da usina de Angra 1 além da sua vida útil inicialmente projetada, é necessário o levantamento de dados de temperatura, dentro outros fatores degradantes, para viabilizar o estudo da vida qualificada desses equipamentos para elaborar corretamente o PQAEE [3].

Portanto, como introduzido no capítulo 1, a fim de mapear as condições ambientais que compõem o espaço de trabalho desses equipamento, a operadora da usina instalou conjuntos de detecção, compostos por sensores de temperatura e radiação, nos pontos internos do prédio de contenção onde esses dispositivos elétricos estão em operação. Este conjunto, chamado de SMOs, fará o levantamento desses dados em todos os novos ciclos de operação até o próximo fim da vida útil estendida, garantindo o monitoramento necessário para a verificação da vida qualificada desses equipamentos em funcionamento previstos no PQAEE, e informando o estado de saúde desses dispositivos permitindo efetuar eventuais substituições[4].

Por outro lado, esse procedimento teve início a partir de 2015, logo, não há dados referentes a períodos anteriores a essa instalação, o que prejudica o processo de qualificação de vida desses equipamentos levando em conta o seu funcionamento desde o início da operação da usina até a data de início dos SMOs.

Entretanto, tal ação de instalação dos medidores começou a ser realizada a partir de 2015, não havendo medições da exposição à temperatura diretamente nos equipamentos específicos do PQAEE, dentro do prédio de contenção da usina, nos anos

anteriores a isso, dificultando que o estudo de envelhecimento e degradação devido a esse fator. Todavia, através do Sistema Integrado de Computadores de Angra 1 (SICA)[17], é possível obter dados de variáveis físicas de toda a operação da usina, especialmente em pontos internos ao prédio de contenção, e como demonstrado por Pinheiro[4] essas variáveis estão correlacionadas fisicamente a temperatura nas posições nos quais os SMOs foram inseridos, permitindo assim, a reconstrução, ou inferência, desses dados ausentes com base nas variáveis monitoradas pelo SICA.

Neste sentido, este trabalho se propõe a estudar diferentes modelos e arquiteturas de redes neurais na inferência desses dados de temperatura referentes ao início de operação da usina até as instalações dos SMOs a partir das variáveis dos SICA e os dados reais monitorados dos SMOs a partir de 2015.

3.2 Modelagem do Problema

Como exposto, o PIT consiste em reconstruir as leituras de sensores SMOs virtuais no período passado até a instalação de 2015. Para isto, os dados reais dos SMOs instalados a partir de 2015 serão correlacionados, a partir de modelos de Redes Neurais, as variáveis do SICA de forma que, após o treinamento, os modelos serão capazes de inferir as temperaturas com base nas variáveis do SICA das datas passadas de forma a obedecer as rela coes físicas entre os dois dados.

3.2.1 Variáveis Estudadas

O procedimento de escolha das variáveis para esse trabalho seguiu-se baseado na mineração de dados feita por Pinheiro, que usando métodos de matriz de correlação e algoritmos de floresta aleatória (do inglês - *Random Forest*) obteve o mapa de calor das correlações de todas as variáveis presente no SICA com os SMOs, identificou as variáveis que mais impactam nessa correlação e extraiu as mais relevantes para o problema. Isso pressuposto, este trabalho se beneficia dos resultados de Pinheiro, abstraindo a etapa de pré-processamento de dados e seleciona da conclusão dos métodos realizados por ele as variáveis mais relevantes e de interesse desse conjunto de trabalho para ser estudada[4].

Portanto, segue abaixo a Tabela 3.1 que apresenta as variáveis selecionadas do SICA e suas respectivas variáveis regressoras dos SMOs com seus respectivos anos

3.2.2 API Functional Keras

Tradicionalmente, no que se refere a construção de modelos de redes neurais, foi usado como principal ferramenta a plataforma de desenvolvimento de modelos de machine learning da biblioteca Keras criada François Chollet, que até o momento

SMO	Variáveis Regressoras	Ano de estudo
SMO06	SMPC02, SMPC55, TE23	2016-2017
SMO08	SMPC02, TE23, TE30	2016-2017
SMO27	SMPC30, TI5703, TI5702	2017-2020
SMO07	TI5702	2017-2020
SMO26	TI5702	2017-2020

Tabela 3.1: Tabela de SMO e Variáveis Regressoras

deste trabalho é a plataforma mais utilizada no desenvolvimento de algoritmos de *Deep Learning*, contemplando todas as arquiteturas, desde modelos MLPs mais simples até Large Language Models (LLMs) [32].

Desde sua criação, a API Sequential foi a mais popular na construção dos modelos e arquiteturas, permitindo abstrair a codificação de cada neurônio e suas conexões, possibilitando aos programadores empilhar as camadas das redes sequencialmente sem precisar dedicar tempo a programação de suas ligações e complexidades matemáticas, o que expandiu o desenvolvimento de novas aplicações de redes neurais nas mais diversas áreas. Entretanto, com o advento dos modelos mais complexos que possuem conexões entre camadas de maneira não sequencial, como as redes Transform[33], LLM[32], etc, foi necessário o desenvolvimento de uma nova ferramenta que permitisse a construção desses modelos neurais não-lineares. Neste sentido, Chollet, em 2019, apresentou a API Functional [7] capaz de construir arquiteturas de redes personalizadas com conexões e fluxo de dados arbitrários, por exemplo, a criação de um modelo com múltiplas camadas de entrada, como mostrado na figura 3.1.

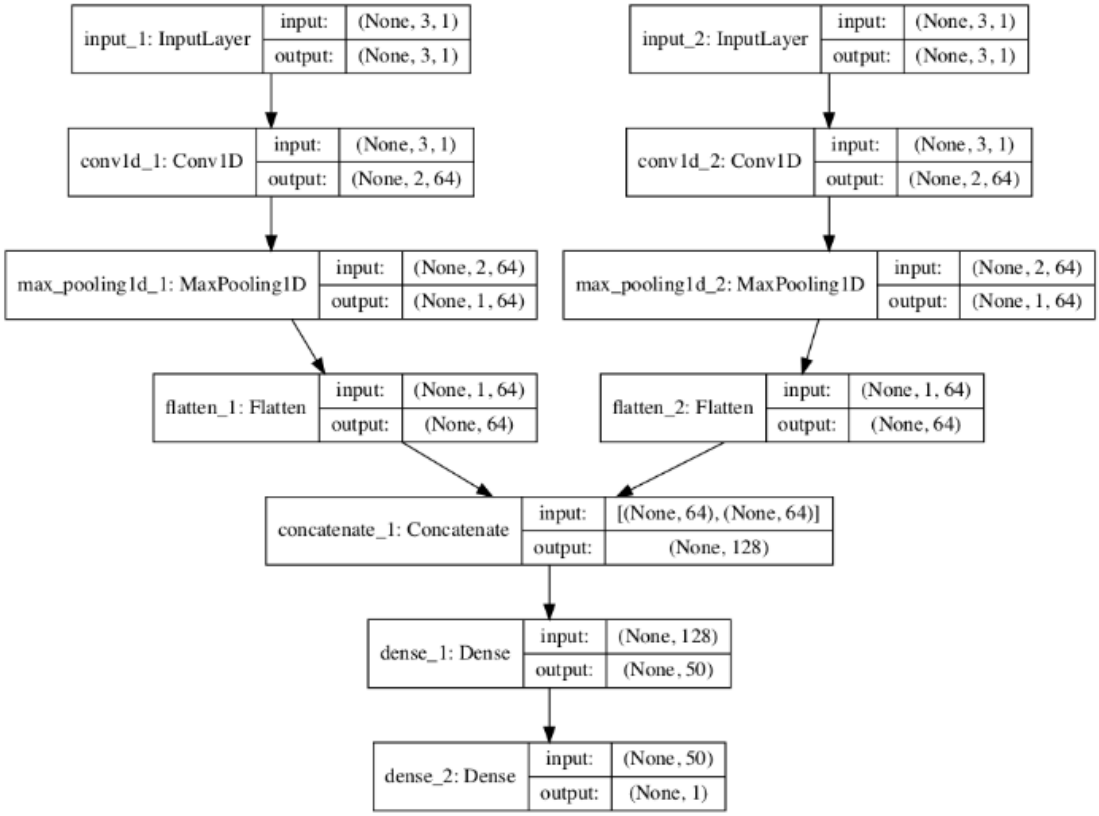

Figura 3.1: Arquitetura de rede arbitrária composta de duas camadas de entrada do tipo Conv1d conectadas a uma saída densa

3.2.3 Modelagem Experimental: Arquiteturas e Variáveis

Através de inúmeros testes, seguindo as boas práticas no que refere ao processo de modelagem de redes neurais, e explorando as funcionalidades e flexibilidade da API Functional, a arquitetura da figura 3.2 foi escolhida para o estudo deste trabalho. O modelo consiste em três camadas paralela de entradas do tipo convolucional de uma dimensão que, após um filtro de *Maxpooling*, são conectadas a um ramo único composto por três camadas sequenciais do tipo LSTM, que por fim se conectam a uma saída densa. Esse mesmo modelo foi utilizado para a regressão de todos os SMOs apresentados na tabela 3.2, e abaixo na tabela 3.2 estão as características dos hiperparâmetros treináveis da rede para cada camada.

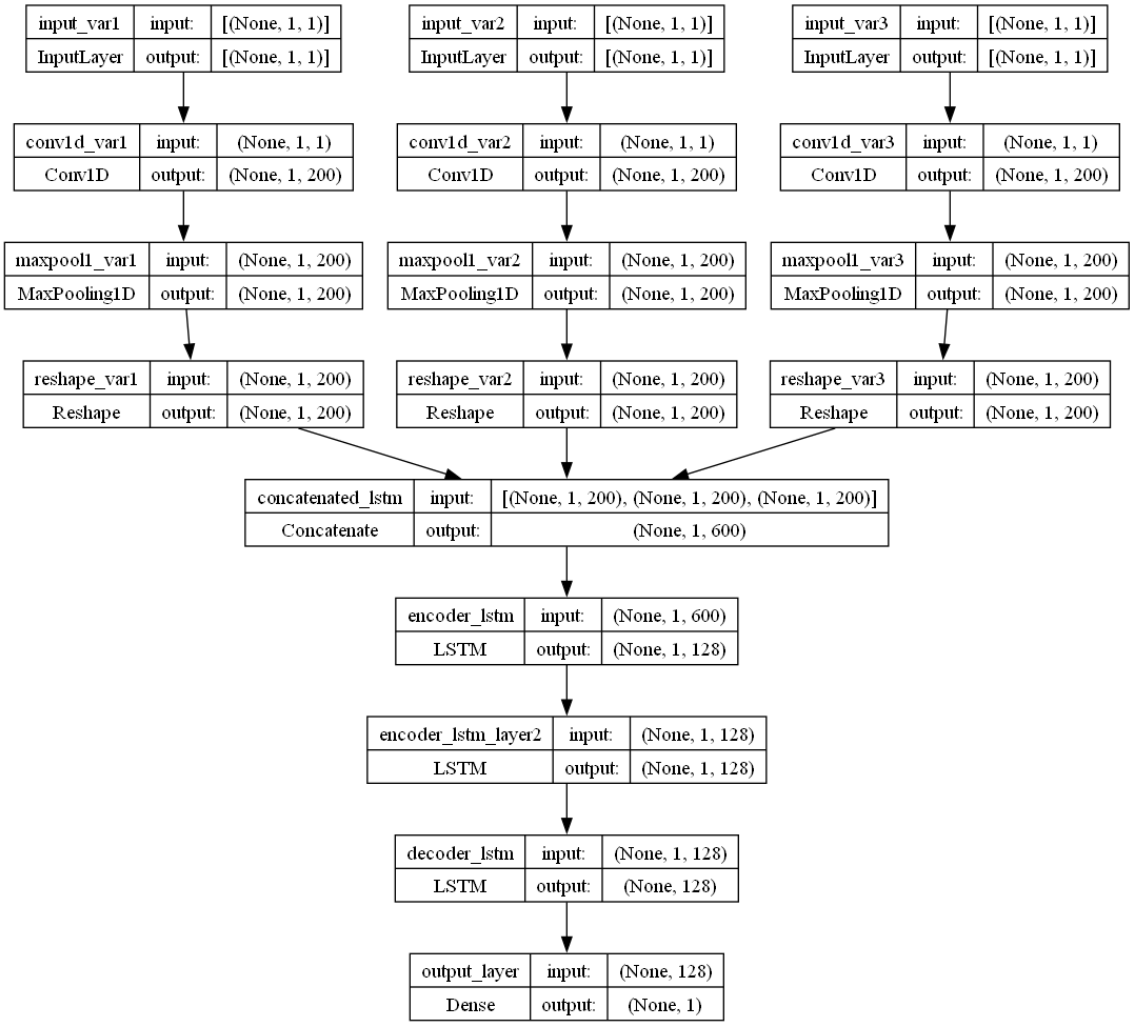

Figura 3.2: Modelo construído para estudo de viabilidade de redes multifacetadas a problema de regressão

Tabela 3.2: Tabela de camadas, número de neurônios, dropout e regularização

CAMADAS	**N neurônios**	**Dropout**	**Regularizacao**
Conv1D_1	200	-	-
Conv1D_2	200	-	-
Conv1D_3	200	-	-
LSTM_1	128	0.2	0.2
LSTM_2	128	0.2	0.2
LSTM_3	128	0.2	0.2
Dense	1	-	-

3.2.4 Treinamento

Seguindo as boas práticas no que tange o processamento de dados, treinamento, validação e teste de redes neurais artificiais; e como apresentado por Pinheiro[4], levando em consideração as variações anuais, com troca de numeração e oscilações na operação dos SMOs, os dados desses foram concatenados a fim de permitir uma sequência de medidas de leitura contínua, de forma que a fase de treinamento permita ao modelo de rede entender as correlações das variáveis regressoras com os SMOs diretamente. Por fim, definiu-se uma separação dos dados em uma proporção de 60% para treino, 20% para validação e 15% para teste, os dos dados de teste são mantidos fora do treinamento, ou seja, a rede tem sua capacidade de reconstrução testada com dados passados no qual ela não viu durante o treino. Além disso, para a análise de desempenho dos modelos é usado como função de perda o MAE que permite verificar o quão perto dos dados de testes a reconstrução alcança.

Nesse sentido, os parâmetros de treinamento utilizados para o estudo do modelo de rede proposto neste trabalho são apresentados na tabela abaixo:

Tabela 3.3: Parâmetros de Treinamento

Otimizador	Adam
Taxa de aprendizado	0.0001
Função de Perda	MAE
Épocas	1000
Batch size	128
Early stopping	400

Além disso, para fins de análise de desempenho do modelo, foram utilizadas duas redes de benchmark já estudadas e aplicadas no PIT pelo Laboratório de Monitoração de Multiprocessos (LMP): uma rede CNN simples e outra LSTM simples, cujas características estão detalhadas nas tabelas 3.4 e 3.5, respectivamente. Nesse contexto, essas redes foram projetadas para simular a ideia de arquiteturas separadas, que serão comparadas ao modelo que combina ambas as redes. Assim, busca-se oferecer uma abordagem adicional para compreender o funcionamento do modelo proposto na Figura 3.1, avaliando se a combinação das duas arquiteturas obtêm um desempenho superior aos modelos mais simples. Por fim, realiza-se um teste de reconstrução simultânea para os SMOs que compartilham variáveis regressoras, conforme indicado na Tabela 3.1. Esse teste tem o objetivo de avaliar a capacidade do modelo de reter conhecimento sobre todas as variáveis e reconstruir, de forma simultânea, todos os SMOs.

Tabela 3.4: Arquitetura de Rede CNN Simples

Camadas	**N Neurônios**	**Dropout**
Conv1D_1	1000	-
Conv1D_2	1000	-
Conv1D_3	1000	-
Dense	512	0.1
Dense	512	0.1
Dense	512	0.1
Dense	1	-

Tabela 3.5: Arquitetura de Rede LSTM Simples

Camadas	**N Neurônios**	**Dropout**
LSTM	1000	0.2
LSTM	1000	0.2
LSTM	1000	0.2
Dense	512	-
Dense	512	-
Dense	512	-
Dense	1	-

Capítulo 4

Resultados e Discussões

Como mencionado no capítulo anterior, a etapa de pré-processamento dos dados se abstrai neste trabalho visto que o processo se dá continuidade a pesquisa publicada por Pinheiro [4]. Logo, de forma mais objetiva, os este capítulo apresenta os resultados obtidos do treinamento dos três modelos de rede apresentados anteriormente, com foco na análise de desempenho do modelo combinado (CNN-LSTM).

Portanto, os resultados do modelo proposto são analisados em conjunto aos resultados das redes de *benchmark*, de forma que fique mais claro a diferença de performance em cada cenário.

Para melhor visualização e entendimento, o treinamento dos modelos será separado em dois tipos, Multivariado e Univariado, seguindo a ordem das variáveis dos SMOs apresentados na tabela 3.1; sendo assim, em dois tópicos:

- **Treinamento Multivariado:**
  - SMO06
  - SMO08
  - SMO27
- **Treinamento Univariado:**
  - SMO07
  - SMO26

Além disso, como pode ser visto através Tabela 3.1, os SMOs 6 e 8 compartilham de variáveis regressoras com uma diferença em apenas nas variáveis SMPC55 e T30. Sendo assim, esta situação é adequada para testar a viabilidade do modelo proposto em expandir a capacidade de reconstruir múltiplas variáveis simultaneamente, necessitando apenas de agregar novas camadas de entrada responsáveis pelas novas variáveis regressoras. Neste sentido, a mesma situação se mostra favorável para teste

com os SMOs 26 e 7 que compartilham a variável regressora TI5702, quer também será avaliado neste capítulo.

Ademais, os parâmetros de análise para os resultados abaixo são o MAE e o comportamento visual do gráfico, visto que, apesar de um resultado promissor do MAE, ou seja, próximo de zero, o resultado pode não condizer com o comportamento esperado da variável.

4.0.1 Treinamento Multivariado

- SMO06

  Analisando os erros apresentados na tabela 4.1 para os três modelos, nota-se que, para fins práticos todos obtiveram a mesma performance para os dados de testes, com uma diferença de 0.98% do modelo combinado para as redes de *benchmark*. No entanto, para os dados de validação, o modelo combinado obteve um resultado melhor se comparado aos outros dois, sendo aproximadamente 50% menor com relação a ambas as redes mais simples.

Tabela 4.1: MAE para a reconstrução para SMO06

Modelo	**CNN**	**LSTM**	**Modelo Combinado**
MAE_treino	0.3647	0.3217	0.3714
MAE_validacao	0.3602	0.3509	0.1813
MAE_teste	0.2777	0.2944	0.2865

  Esta performance semelhante se reflete no gráfico da figura 4.1. Portanto, é possível considerar que os três modelos obtiveram resultados satisfatórios na reconstrução dos dados para o SMO06.

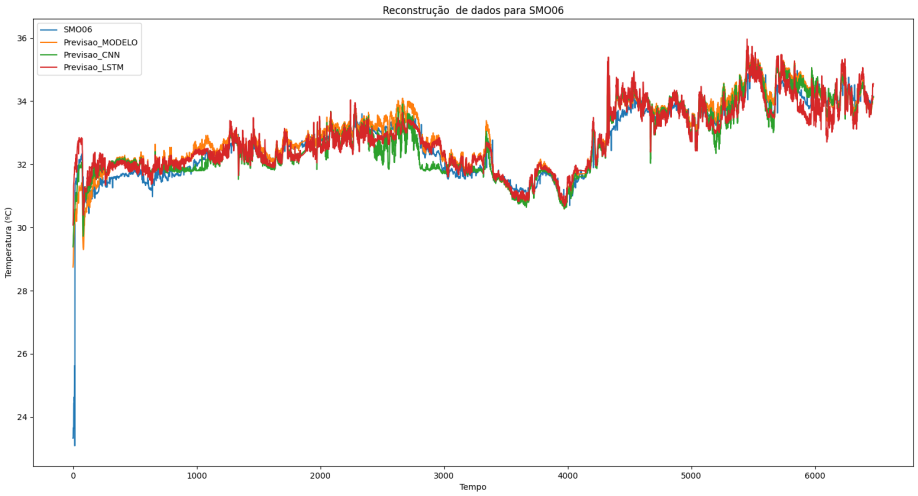

Figura 4.1: Resultado da reconstrução dos dados para SMO06

- SMO08

  Pelo dados da tabela 4.2 verifica-se um resultado semelhante ao caso anterior onde todos tiveram o mesmo desempenho. No entanto, apesar desse resultado ser considerado satisfatório com relação ao parâmetro de erro médio absoluto, nota-se que o mesmo para esse caso é ligeiramente maior. Além disso, esse aumento é notado no gráfico da figura 4.2, onde, pela semelhança do formato entre os dois SMOs, é perceptível a diferença dos dados previstos dos dados reais

Tabela 4.2: MAE para a reconstrução para SMO08

Modelo	**CNN**	**LSTM**	**Modelo COMB**
MAE_treino	0.2934	0.2866	0.3462
MAE_validacao	0.3354	0.2326	0.2427
MAE_teste	0.8705	0.6227	0.6982

Verifica-se também o desempenho superior da rede LSTM mais simples com relação aos outros dois modelos obtendo um erro em torno de 11.19% menor com relação ao modelo combinado e 28.47% menor que a rede CNN.

Figura 4.2: Resultado da reconstrução dos dados para SMO06

- SMO27

No seguinte caso nota-se, para todos os modelos uma piora no desempenho de forma considerável com relação aos casos anteriores. Além disso, apesar do modelo combinado obter o menor erro para os dados de teste, seu erro durante o treinamento é 6.04 vezes maior que o modelo LSTM e 7.53 vezes o valor da rede CNN. Portanto,embora os resultados possam indicar que o modelo combinado possui uma capacidade melhor de generalização para esse caso, um estudo mais aprofundado nas métricas de treinamento, como novos hiperparâmetros, tempo de treinamento e separação de dados é necessário para uma conclusão precisa.

Tabela 4.3: MAE para a reconstrução para SMO27

Modelo	**CNN**	**LSTM**	**Modelo Comb**
MAE_treino	0.3127	0.3898	2.3556
MAE_validacao	1.9830	1.9649	1.4796
MAE_teste	3.4905	3.3778	3.0857

Por outro lado, ao se analisar os resultados visualmente pelo gráfico da figura 4.3, fica evidente que nenhum dos modelos conseguiu reproduzir com fidelidade os dados de teste. Pode-se observar o erro menor do modelo combinado demonstrado pelo gráfico, onde o modelo se ajusta melhor a tendência de movimento da série temporal. No entanto, para este cenário, nenhum modelo obteve um resultado satisfatório.

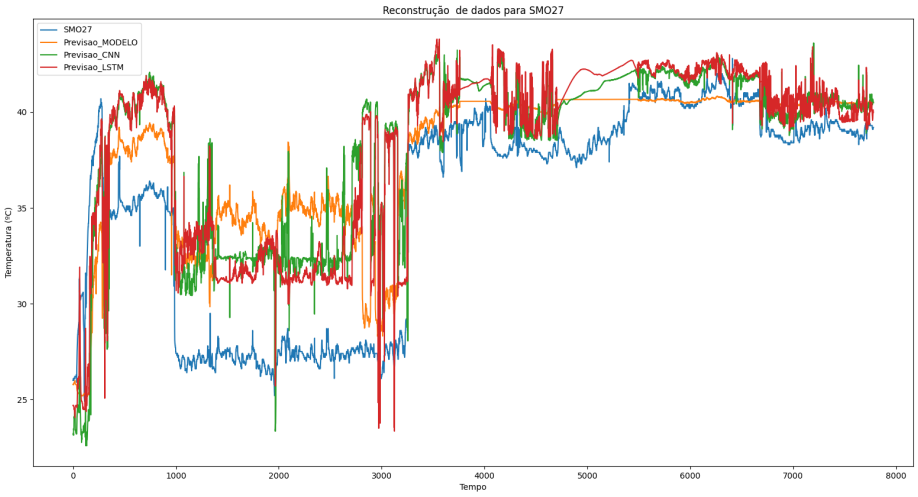

Figura 4.3: Resultado da reconstrução dos dados para SMO27

4.0.2 Treinamento Univariado

- SMO07

  Neste cenário, pelos resultados da tabela 4.4, pode-se afirmar que os três modelos obtiveram o mesmo desempenho com os modelos de *benchmark* obtendo praticamente o mesmo resultado e o modelo combinado uma diferença de apenas 3%.

Tabela 4.4: MAE para a reconstrução para SMO07

Modelo	**CNN**	**LSTM**	**Modelo Comb**
MAE_treino	3.6151	3.6110	4.2220
MAE_validacao	4.0927	4.1577	3.6520
MAE_teste	2.8915	3.2731	2.7885

Ao observar o gráfico da figura 4.4 verifica-se que o erro baixo (menor que 10%) se reflete apenas no fato da distribuição dos dados previstos pela rede estar dentro do da região dos dados reais; em outras palavras para um mesmo dado previsto que ultrapassa um valor real positivamente, outro valor está abaixo na mesma intensidade, levando a um MAE baixo, pois o erros dessas duas situações se contrapõem, montando um valor médio relativamente baixo. Neste contexto, tanto este cenário quanto o anterior demonstram que uma análise puramente quantitativa do desempenho de redes neurais através dos erros pode prejudicar a compreensão do que de fato está ocorrendo.

Além disso, nota-se pelo gráfico que o modelo combinado se manteve mais estável e teve menos flutuações, o que para esse cenário não é satisfatório após o primeiro vale da curva, onde evidencia que essa estabilidade impediu o modelo se ajustar aos picos dessas região, reproduzindo uma curva dentro da média dos pontos, provando o caso indesejável explicado no parágrafo anterior.

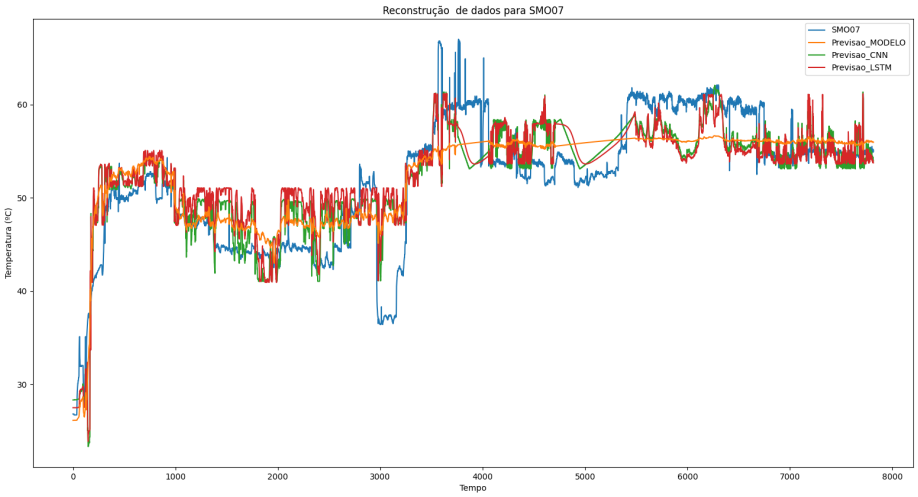

Figura 4.4: Resultado da reconstrução dos dados para SMO07

- SMO26

  O cenário apresentado pela tabela 4.1 representa uma situação semelhante ao SMO07. No entanto houve uma melhora no desempenho para todos os modelos, com o modelo combinado obtendo resultados semelhantes às redes de *benchmark* com uma diferença de 5.2% da rede LSTM e 4.2% da CNN.

Tabela 4.5: MAE para a reconstrução para SMO26

Modelo	**CNN**	**LSTM**	**Modelo Comb**
MAE_treino	1.1529	1.1718	1.2212
MAE_validacao	1.6703	1.6846	1.5763
MAE_teste	1.3168	1.3039	1.3729

Através de análise qualitativa pelo gráfico 4.5, verifica-se que os três modelos se ajustaram ao primeiro vale da curva, enquanto que, por outro lado o modelo combinado mantém a rigidez na variação dos dados após o vale, o impedindo de se ajustar as flutuações do dado real. Além disso as redes simples repetem esse comportamento.

Uma das possíveis razões para esse fenômeno tanto no SMO26 quanto SMO07 é que a única variável regressora, apesar de ter uma correlação aos dados dos SMOs, não é suficiente para descrever o comportamento desses equipamentos. Neste sentido, pode haver alguma variável não identificada que o comportamento esteja mais diretamente correlacionada ao SMOs.

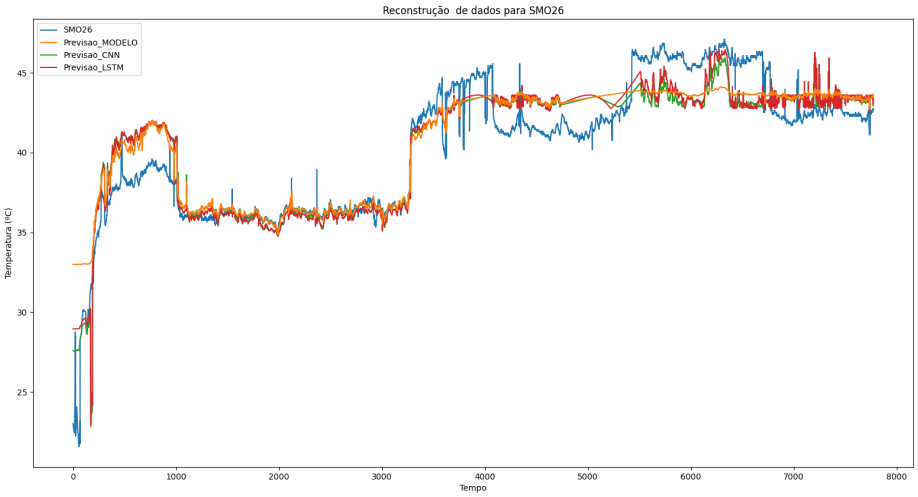

Figura 4.5: Resultado da reconstrução dos dados para SMO26

4.0.3 Treinamento Simultâneo

- SMO06 - SMO08

  Para o primeiro teste de reconstrução simultâneoa fim de validar a hipótese de um modelo único regressor é feito utilizando as variáveis SMPC22, SMPC55, T23 e T30 para a reconstrução dos SMO06 e SMO08, onde o modelo utilizou das variáveis semelhantes entre esses SMOs, com acréscimo de uma nova camada de entrada Conv1D para a variável T30. Por fiom, nota-se pela Tabela 4.6, e comparando com os resultados dos testes individuais para ambos os SMOs apresentados nas tabelas 06 e 08, que esta aplicação obteve resultados satisfatórios, visto que apesar dos erros de validação e treinamento serem maiores, o erros nos dados de teste foram menores, com uma redução 4.5% para SMO06 e 60.22% para o SMO08, evidenciando uma capacidade maior de generalização dos dados nessa nova configuração.

	SMO06	SMO08
MAE Treino	1.0465	1.0678
MAE Validação	0.9759	1.0229
MAE Teste	0.2736	0.2777

Tabela 4.6: MAE para a reconstrução simultânea de SMO06 e SMO08

Além disso, pelo gráfico da Figura 4.7 é possível observar com nitidez a diferença de performance para o SM08 onde verifica-se um ajuste mais fino e próximo dos dados reais em comparação ao gráfico anterior apresentado na figura 4.2, provando assim a eficiência do modelo em reconstruir dados de múltiplos SMOs.

Figura 4.6: Resultado da reconstrução simultânea pra o SMO06

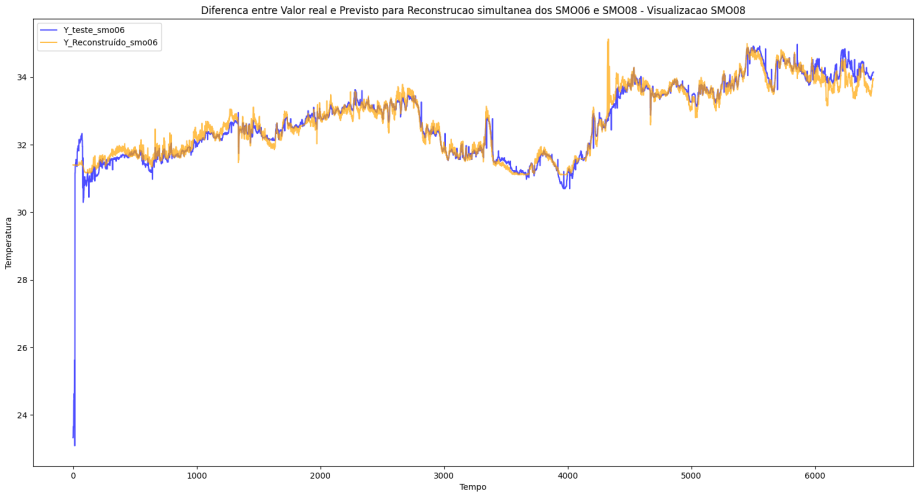

Figura 4.7: Resultado da reconstrução simultânea pra o SMO08

- SMO26 - SMO07

Para o cenário dos SMO26 e SMO07, que possuem apenas uma única variável regressora (TI5072) a reconstrucao simultanea se manteve dentro dos mesmos resultados apresentados nos testes individuais, como pode ser visto ao comparar os dados das Tabelas 4.5 e 4.4 com os resultados do novo teste apresentado na Tabela 4.7. Bem como ao comparar os gráficos para ambos os testes individuais, vide Figuras 4.5 e 4.4, com o gráfico da reconstrução em paralelo abaixo apresentado na Figura 4.8, verifica-se que o modelo manteve o mesmo comportamento, não havendo melhora significativa nos resultados ou perda desempenho ao adicionar uma variável para ser reconstruída.

	SMO26	**SMO07**
MAE Treino	1.2502	3.9970
MAE Validação	1.6746	3.3689
MAE Teste	1.4009	3.2589

Tabela 4.7: MAE para a reconstrução simultânea de SMO26 e SMO07

Por outro lado, o uso de um único modelo para reconstruir múltiplas variáveis se mostra útil no que tange a redução de modelos treináveis necessários para alcançar certo resultado, apesar de não haver melhora nesse cenário especifico. Além disso, esse resultado reforça a hipótese apresentada anteriormente, que para os cenários de teste univariado a variável não seja suficiente para per-

mitir o modelo construir a informação a cerca do comportamento do SMOs, tanto em teste individual quanto simultâneo,

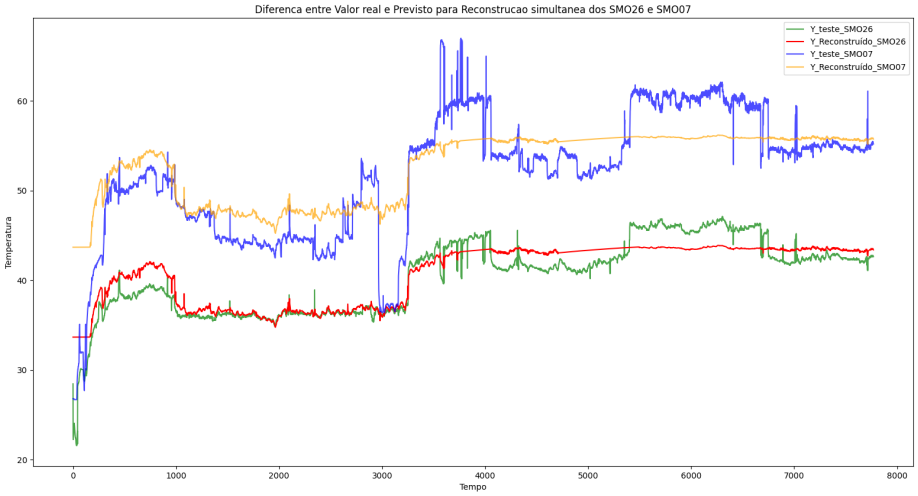

Figura 4.8: Resultado da reconstrução simultânea pra os SMO26 e SMO07

Capítulo 5

Conclusão

Desde o renascimento das redes neurais, após o período conhecido como "inverno das redes neurais", o número crescente de estudos sobre aplicações desses modelos e algoritmos tem ganhado destaque no meio acadêmico. O desenvolvimento contínuo de tecnologias avançadas (como novas arquiteturas, otimizadores e modelos) amplia as possibilidades de aplicação em diversas áreas. No contexto da engenharia nuclear, essas tecnologias oferecem inúmeras oportunidades promissoras.

Nesse cenário, o estudo das redes neurais aplicado à inferência de temperaturas, embora ainda em estágio inicial, mostra-se promissor, como evidenciado pelos resultados de Pinheiro e Cardoso. Além disso, este trabalho, por sua vez, propõe uma nova abordagem para a construção de arquiteturas não lineares, demonstrando que os modelos tradicionais continuam relevantes no atual panorama competitivo com a utilização da API Functional do Keras que permite a criação de modelos híbridos e altamente flexíveis.

Com relação aos resultados obtidos pelo modelo proposto neste trabalho, destaca-se a melhora no desempenho geral para todas as variáveis dos SMOs reconstruídas. No entendo, nas variáveis SMO27, SMO26 e SMO07, o desempenho, apesar de superior com relação às duas arquiteturas tradicionais CNN e LSTM, com uma análise mais detalhada aos gráficos de reconstrução, nota-se que o modelo combinado não se distanciou muito dos resultados dos tradicionais. Apesar do modelo ter "compreendido" a tendência de movimento dos dados para os SMO07 e SMO26, ele não obteve uma resolução suficiente para reconstruir os dados de forma precisa. Enquanto que, para os SMO27 e SMO03 o modelo não foi capaz de assimilar tanto o movimento quanto às amplitudes dos dados.

Portanto, como todos os modelos estudados não obtiveram resultados satisfa-

tórios para as variáveis SMO27, SMO26 e SMO07. É aceitável admitir que, apesar de haver uma correlação entre as variáveis regressoras e esses SMOs, há a possibilidade de uma variável que se relacione diretamente com as regressoras e SMOs que não esteja presente no conjunto de dados, ou seja, as variáveis regressoras utilizadas não são suficientes para a reconstrução desses SMOs, ainda que consigam expressar a tendência.

Por outro lado, para os SMO06 e SMO08, o modelo combinado obteve resultados satisfatórios e com relação às redes de *benchmark*, evidenciando que, a princípio, a sua utilização é promissora, no entanto não justificável em casos no qual o problema concentra-se apenas uma única variável a ser estudada.

Por fim, o resultado apresentado para o teste de reconstrução simultânea para os SMO06 e SMO08, com um desempenho 60,22% para o SMO08, prova a viabilidade dessa arquitetura em se tornar um modelo multi-propósito capaz de reconstruir múltiplas variáveis simultaneamente, e enquanto que o teste realizado com os SMOs 26 e 07 mostram que apesar de não haver um ganho de desempenho nesse cenário, o modelo ainda se mostra útil no que diz respeito a manter a performance com menos recurso e necessidade de múltiplos treinamentos para cada variável.

Referências Bibliográficas

[1] KHARECHA, P. A., HANSEN, J. E. "Prevented mortality and greenhouse gas emissions from historical and projected nuclear power", *Environmental science & technology*, v. 47, n. 9, pp. 4889–4895, 2013.

[2] DE PESQUISA ENERGÉTICA, E. E. "Matriz Energética e Elétrica". 2022. Disponível em: <`https://www.epe.gov.br/pt/abcdenergia/matriz-energetica-e-eletrica`>. Acessado em: 10 de agosto de 2024.

[3] CARDOZO, F. H. P. *APLICAÇÃO DE REDES NEURAIS PARA INFERÊNCIA DE TEMPERATURAS HISTÓRICAS NA USINA NUCLEAR ANGRA*. Tese de Doutorado, Universidade Federal do Rio de Janeiro, 2022.

[4] PINHEIRO, V. H. C. "Redes neurais de aprendizado profundo aplicadas a problemas complexos da engenharia nuclear", 2020.

[5] ELLER, I. B., OTHERS. "Uma metodologia para avaliação regulatória de extensão de vida de usinas nucleares", 2018.

[6] PINHEIRO, V. H. C., DOS SANTOS, M. C., DO DESTERRO, F. S. M., et al. "Nuclear Power Plant accident identification system with "don't know" response capability: Novel deep learning-based approaches", *Annals of nuclear energy*, v. 137, pp. 107111, 2020.

[7] CHOLLET, F. "The Functional API". 2023. Disponível em: <`https://keras.io/guides/functional_api/`>. Acessado em: 30 de agosto de 2024.

[8] MCCULLOCH, W., PITTS, W. "A Logical Calculus of the Ideas Immanent in Nervous Activity", *The Bulletin of Mathematical Biophysics*, v. 5, n. 4, pp. 115–133, 1943. doi: 10.1007/BF02478259.

[9] HEBB, D. O. *The Organization of Behavior: A Neuropsychological Theory*. New York, John Wiley & Sons, 1949.

[10] ROSENBLATT, F. "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain", *Psychological Review*, v. 65, n. 6, pp. 386–408, 1958. doi: 10.1037/h0042519.

[11] RUMELHART, D. E., HINTON, G. E., WILLIAMS, R. J. "Learning representations by back-propagating errors", *Nature*, v. 323, n. 6088, pp. 533–536, 1986. doi: 10.1038/323533a0.

[12] HOPFIELD, J. J. "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", *Proceedings of the National Academy of Sciences of the United States of America*, v. 79, n. 8, pp. 2554–2558, 1982. doi: 10.1073/pnas.79.8.2554.

[13] BENGIO, Y., SIMARD, P., FRASCONI, P. "Learning long-term dependencies with gradient descent is difficult", *IEEE Transactions on Neural Networks*, v. 5, n. 2, pp. 157–166, 1994. doi: 10.1109/72.279181.

[14] HOCHREITER, S. "Long Short-term Memory", *Neural Computation MIT-Press*, 1997.

[15] LECUN, Y., BOTTOU, L., BENGIO, Y., et al. "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, v. 86, n. 11, pp. 2278–2324, 1998. doi: 10.1109/5.726791.

[16] KARIM, F., MAJUMDAR, S., DARABI, H., et al. "Multivariate LSTM-FCNs for time series classification", *Neural networks*, v. 116, pp. 237–245, 2019.

[17] DOS SANTOS, M. C. *SISTEMA BASEADO EM REDES NEURAIS CONVOLUCIONAIS, RECORRENTES E AUTOCODIFICADORAS PARA CLASSIFICAÇÃO DE ACIDENTES POSTULADOS EM CENTRAIS NUCLEARES COM CAPACIDADE DE DETECÇÃO DE ANOMALIAS E RESPOSTA "NÃO SEI"*. Tese de Doutorado, Universidade Federal do Rio de Janeiro, 2023.

[18] WANG, X., LIU, Y., XIN, H. "Bond strength prediction of concrete-encased steel structures using hybrid machine learning method". In: *Structures*, v. 32, pp. 2279–2292. Elsevier, 2021.

[19] RAMOS, R. H., DIAS, M. D. P. *Sistema de contagem, classificação de grupo de idade e segmentação de vestimenta de pessoas em vídeos utilizando deep learning.* B.S. thesis, Universidade Tecnológica Federal do Paraná, 2022.

[20] AFAN, H. A., IBRAHEM AHMED OSMAN, A., ESSAM, Y., et al. "Modeling the fluctuations of groundwater level by employing ensemble

deep learning techniques", *Engineering Applications of Computational Fluid Mechanics*, v. 15, n. 1, pp. 1420–1439, 2021.

[21] HAYKIN, S. *Neural networks and learning machines, 3/E*. Pearson Education India, 2009.

[22] GOODFELLOW, I., BENGIO, Y., COURVILLE, A. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[23] DUCHI, J., HAZAN, E., SINGER, Y. "Adaptive subgradient methods for online learning and stochastic optimization." *Journal of machine learning research*, v. 12, n. 7, 2011.

[24] DIEDERIK, P. K. "Adam: A method for stochastic optimization", *(No Title)*, 2014.

[25] SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., et al. "Dropout: a simple way to prevent neural networks from overfitting", *The journal of machine learning research*, v. 15, n. 1, pp. 1929–1958, 2014.

[26] LECUN, Y., BOSER, B., DENKER, J. S., et al. "Backpropagation applied to handwritten zip code recognition", *Neural computation*, v. 1, n. 4, pp. 541–551, 1989.

[27] YAMASHITA, R., NISHIO, M., DO, R. K. G., et al. "Convolutional neural networks: an overview and application in radiology", *Insights into imaging*, v. 9, pp. 611–629, 2018.

[28] MU, R. "A survey of recommender systems based on deep learning", *Ieee Access*, v. 6, pp. 69009–69022, 2018.

[29] ZHANG, J., LUO, F., QUAN, X., et al. "Improving wave height prediction accuracy with deep learning", *Ocean Modelling*, v. 188, pp. 102312, 2024. ISSN: 1463-5003. doi: https://doi.org/10.1016/j.ocemod.2023.102312. Disponível em: <`https://www.sciencedirect.com/science/article/pii/S146350032300152X`>.

[30] SMAGULOVA, K., JAMES, A. P. "A survey on LSTM memristive neural network architectures and applications", *The European Physical Journal Special Topics*, v. 228, n. 10, pp. 2313–2324, 2019.

[31] DAVID, P., MONTANARI, G. "Compensation effect in thermal aging investigated according to Eyring and Arrhenius models", *European Transactions on Electrical Power*, v. 2, n. 3, pp. 187–194, 1992.

[32] BROWN, T., MANN, B., RYDER, N., et al. "Language models are few-shot learners", *Advances in neural information processing systems*, v. 33, pp. 1877–1901, 2020.

[33] VASWANI, A. “Attention is all you need”, *Advances in Neural Information Processing Systems*, 2017.