



SISTEMA BASEADO EM REDES NEURAIAS CONVOLUCIONAIS,
RECORRENTES E AUTOCODIFICADORAS PARA CLASSIFICAÇÃO DE
ACIDENTES POSTULADOS EM CENTRAIS NUCLEARES COM CAPACIDADE
DE DETECÇÃO DE ANOMALIAS E RESPOSTA “NÃO SEI”

Marcelo Carvalho dos Santos

Tese de Doutorado 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 Doutor em Engenharia Nuclear.

Orientadores: Prof. Roberto Schirru, D.Sc.

Prof. Cláudio Márcio do Nascimento

Abreu Pereira, D.Sc.

Rio de Janeiro

Abril de 2023

SISTEMA BASEADO EM REDES NEURAIAS CONVOLUCIONAIS,
RECORRENTES E AUTOCODIFICADORAS PARA CLASSIFICAÇÃO DE
ACIDENTES POSTULADOS EM CENTRAIS NUCLEARES COM CAPACIDADE
DE DETECÇÃO DE ANOMALIAS E RESPOSTA “NÃO SEI”

Marcelo Carvalho dos Santos

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

Orientadores: Roberto Schirru

Cláudio Márcio do Nascimento Abreu Pereira

Aprovada por: Prof. Roberto Schirru

Prof. Cláudio Márcio do Nascimento Abreu Pereira

Prof. Paulo Fernando Ferreira Frutuoso e Melo

Prof^a. Andressa dos Santos Nicolau

Prof. César Marques Salgado

Prof. Paulo Victor Rodrigues de Carvalho

RIO DE JANEIRO, RJ – BRASIL

ABRIL DE 2023

Santos, Marcelo Carvalho

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” / Marcelo Carvalho dos Santos. – Rio de Janeiro: UFRJ/COPPE, 2023.

XIV, 88 p.: il.; 29,7 cm.

Orientadores: Roberto Schirru

Claudio Marcio do Nascimento Abreu Pereira

Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia Nuclear, 2023.

Referências Bibliográficas: p. 76-88.

1. Engenharia nuclear. 2. Aprendizado profundo. 3. Engenharia de fatores humanos. I. Schirru, Roberto *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Nuclear. III. Título.

“As palavras mais importantes que uma pessoa pode dizer são 'eu serei melhor' ... Uma jornada terá dor e falhas. Não devemos apenas aceitar os próximos passos. São os tropeços. As tentativas. O conhecimento de que iremos falhar. De que iremos machucar àqueles próximos a nós. Mas se pararmos, se aceitarmos a pessoa que somos quando caímos, a jornada acaba. A falha se torna nosso destino. Amar a jornada é não aceitar esse fim. Eu descobri, através de experiências dolorosas, que o passo mais importante que uma pessoa pode dar é o próximo passo.”

Brandon Sanderson, Oathbringer

AGRADECIMENTOS

Agradeço à minha família, em especial à minha mãe e minha madrinha. Ao longo da minha vida, vocês estiveram sempre presentes, oferecendo amor, apoio e dedicação incondicionais. Suas palavras de incentivo e exemplo de vida me inspiraram a seguir em frente e a nunca desistir dos meus sonhos. Se não fosse pelo suporte incansável de vocês, eu não teria chegado até aqui.

Agradeço aos meus estimados orientadores, Roberto Schirru e Claudio Márcio, por terem confiado em mim e me oferecido a oportunidade de trilhar esta jornada. Suas valiosas orientações, sugestões, críticas e vastos conhecimentos foram fundamentais para a realização deste trabalho e para o meu desenvolvimento acadêmico, profissional e pessoal. Agradeço pelo comprometimento dedicado, incentivo e suporte constante, que ultrapassaram os limites deste trabalho, mas foram valiosos durante toda a minha trajetória acadêmica. Suas orientações me forneceram a base para desenvolver habilidades importantes, que me permitiram crescer não só como profissional, mas como ser humano.

Agradeço a todos os profissionais, professores e pesquisadores do Laboratório de Monitoração de Processos (LMP), que contribuíram de alguma forma para o desenvolvimento deste trabalho. Suas orientações, suporte e sugestões foram fundamentais para a realização desta pesquisa. Além disso, agradeço pela disponibilidade em compartilhar seus conhecimentos e pela paciência em tirar minhas dúvidas.

Agradeço aos meus amigos que me acompanharam durante toda essa jornada, sempre apoiando e encorajando em todos os momentos. Em especial, gostaria de agradecer a Filipe Santana, Juliane Carvalho, Marcos Antônio, Murilo Menezes, Rian Miranda e Victor Pinheiro, cujas palavras de incentivo e encorajamento foram essenciais para me manter motivado durante o processo de elaboração desta tese. A presença de cada um em minha vida foi um verdadeiro presente e sou profundamente grato pelo suporte que me forneceram de formas tão diversas.

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

SISTEMA BASEADO EM REDES NEURAIAS CONVOLUCIONAIS,
RECORRENTES E AUTOCODIFICADORAS PARA CLASSIFICAÇÃO DE
ACIDENTES POSTULADOS EM CENTRAIS NUCLEARES COM CAPACIDADE
DE DETECÇÃO DE ANOMALIAS E RESPOSTA “NÃO SEI”

Marcelo Carvalho dos Santos

Abril/2023

Orientadores: Roberto Schirru

Cláudio Márcio do Nascimento Abreu Pereira

Programa: Engenharia Nuclear

As usinas nucleares são sistemas complexos monitorados por uma equipe de operadores altamente treinados, que em caso de ocorrência de um evento anômalo na usina, como um transiente ou acidente, deve analisar rapidamente um grande número de variáveis para identificar o evento. No entanto, devido ao grande volume de informações que devem ser analisadas rapidamente nessas situações, podem ocorrer erros de identificação. Portanto, este estudo propõe o desenvolvimento de um sistema de identificação de transientes e acidentes em usinas nucleares, estruturado modularmente, baseado em aprendizado profundo, incluindo detecção de anomalias e capacidade de resposta “Não Sei”. Para avaliar a capacidade do sistema, foram conduzidos experimentos utilizando um estudo de caso realista utilizando 16 estados operacionais da usina nuclear Angra 2, do tipo PWR. Como resultado, a melhor configuração de modelos de aprendizado profundo para o sistema alcançou uma exatidão de 100% em termos de identificação correta de resposta “Não Sei”, mantendo uma exatidão de 99,82% na identificação correta de eventos.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

SYSTEM BASED ON CONVOLUTIONAL, RECURRENT AND
AUTOENCODER NEURAL NETWORKS FOR CLASSIFICATION OF
POSTULATED ACCIDENTS IN NUCLEAR POWER PLANTS WITH ANOMALY
DETECTION AND “DON'T KNOW” RESPONSE CAPABILITY

Marcelo Carvalho dos Santos

Abril/2023

Advisors: Roberto Schirru

Cláudio Márcio do Nascimento Abreu Pereira

Department: Nuclear Engineering

Nuclear power plants (NPPs) are complex systems monitored by a team of highly trained operators, which in case of an anomalous event on the NPP, such as a transient or accident, must quickly analyze a high number of variables in order to identify the event. However, due to the great volume of information that must be rapidly analyzed in those situations, misidentifications can occur. Therefore, this study proposes the development of a modular structured, deep learning (DL) based, NPP accident identification system including anomaly detection and “don't know” response capability. In order to assess the system capacity, experiments were conducted using a realistic study case using 16 operational states of the Angra 2 nuclear power plant, PWR type. As a result, the best configuration of DL models for the system achieved an accuracy of 100% in terms of correct “don't know” identification, while still retaining the accuracy of 99.82% on correct event identifications.

SUMÁRIO

INTRODUÇÃO.....	1
1.1. Apresentação do Problema.....	1
1.2. Objetivos.....	4
FUNDAMENTAÇÃO TEÓRICA.....	7
2.1. Revisão Bibliográfica das Técnicas de Inteligência Artificial Aplicadas a de Identificação e Diagnóstico de Acidentes Nucleares.....	7
2.1.1. Técnicas Clássicas de Inteligência Artificial.....	7
2.1.2. Arquiteturas Clássicas de Redes Neurais Artificiais.....	8
2.1.3. Arquiteturas Profundas de Redes Neurais Artificiais.....	9
2.2. Contexto Histórico das Redes Neurais Artificiais.....	10
2.2.1. Era da Cibernética.....	11
2.2.2. Era do Conexionismo.....	12
2.2.3. Era do Aprendizado Profundo.....	14
2.3. Principais Arquiteturas de Redes Neurais.....	16
2.3.1. Redes Neurais Perceptron Multicamadas.....	16
2.3.2. Redes Neurais Convolucionais.....	20
2.3.3. Redes Neurais Recorrentes.....	31
2.3.4. Redes Neurais Autocodificadoras.....	35
DESCRIZAÇÃO DO SISTEMA PROPOSTO.....	38
3.1. Descrição dos Principais Problemas Enfrentados pelo Sistema Proposto.....	38
3.1.1. Classificação de Séries Temporais.....	38
3.1.2. Detecção de Novidade.....	42
3.2. Descrição do Sistema.....	43
3.2.1. Protótipo da Interface do Sistema Proposto.....	44
3.3. Arquiteturas de Redes Neurais Profundas Analisadas para o Sistema Proposto.....	47
3.3.1. Rede Neural Retificada Profunda.....	47
3.3.2. Rede Totalmente Convolucional.....	48
3.3.3. Long Short-Term Memory.....	49
3.3.4. Long Shot-Term Memory-Fully Convolution Network.....	51
3.3.5. LSTM Autoencoder.....	54
3.4. Materiais e Métodos.....	55
ESTUDO DE CASO, EXPERIMENTOS E RESULTADOS.....	56
4.1. Apresentação do Estudo de Caso.....	56
4.1.1. Descrição do Conjunto de Dados.....	56
4.1.2. O Método de Aumento de Dados.....	57

4.1.3. A Técnica de Janela Deslizante.....	58
4.1.4. Modelagem e Distribuição do Conjunto de Dados.....	59
4.2. Experimentos para o Módulo de Classificação de Acidentes Postulados.....	62
4.2.1. Resultados.....	63
4.3. Experimentos para o Módulo de Validação.....	68
3.5.1. Resultados.....	69
CONCLUSÕES.....	74
REFERÊNCIAS BIBLIOGRÁFICAS.....	76

LISTA DE FIGURAS

Figura 1. Estruturas básica de uma MLP (RASCHKA & MIRJALILI, 2017)	17
Figura 2. Esquerda: Uma MLP simples com uma camada oculta. Direita: Uma DFNN com 4 camadas ocultas (VÁZQUEZ, 2017).....	18
Figura 3. Função sigmoide logística e sua derivada (FACURE, 2017)	20
Figura 4. Exemplo uma imagem colorida sendo representada pela camada de entrada de uma CNN como grade 3D (3 matrizes de pixels empilhadas) (SAYEED <i>et al.</i> , 2020).....	21
Figura 5. Esquerda: estrutura interna das camadas de uma MLP. Direita: estrutura interna das camadas de uma CNN (STANFORD, s.d).	22
Figura 6. Descrição do funcionamento do córtex visual (HERZOG & CLARKE, 2014)	23
Figura 7. Representação gráfica da operação $h(x) = f(x) * g(x)$ (BRACEWELL, 1999)	24
Figura 8. Etapas no processo de convolução de dois vetores (RASCHKA & MIRJALILI, 2017)	25
Figura 9. Operação de convolução 2D entre a entrada e a matriz rotacionada (RASCHKA & MIRJALILI, 2017).....	27
Figura 10. Processo de geração de uma mapa de características, através da convolução do filtro com a entrada (YAMASHITA <i>et al.</i> , 2018)	28
Figura 11. Exemplos dos resultados ao convoluir diferentes filtros com uma imagem (ALBAWI, 2018)	29
Figura 12. Esquerda: Operação de max pooling. Direita: Operação de average pooling (MUHAMAD <i>et al.</i> , 2019).....	31
Figura 13. As categorias de dados sequencias (VASILEV <i>et al.</i> , 2019).....	33
Figura 14. Fluxo de Informação em uma RNN (VASILEV <i>et al.</i> , 2019)	34
Figura 15. Arquitetura de autoencoder clássico (SAGHEER & KOTB, 2019).....	36
Figura 16. Evolução temporal de 16 variáveis de estado durante um acidente BLACKOUT em um PWR (De autoria própria).....	39
Figura 17. Evolução temporal de 16 variáveis de estado durante um acidente LOCA (De autoria própria).	40
Figura 18. Estrutura do sistema proposto de identificação e diagnóstico de acidentes nucleares (De autoria própria)	44

Figura 19. Interface do Sistema em Estado Operacional Normal, em Acidente Postulado e em Resposta “Não Sei” (De autoria própria).....	46
Figura 20. Função ReLu e sua derivada (FACURE, 2017).....	47
Figura 21. Célula LSTM (OLAH, 2015).....	51
Figura 22. Estrutura interna da LSTM-FCN (KARIM <i>et al.</i> , 2017).	51
Figura 23. Estrutura interna da Multivariate LSTM-FCN (KARIM <i>et al.</i> , 2019).....	53
Figura 24. Estrutura do LSTM-Autoencoder (SAGHEER & KOTB, 2019)	55
Figura 25. Exemplo da técnica de janela deslizante (De autoria própria).....	59
Figura 26. Evolução do Desempenho da LSTM, LSTM-FCN e FCN em relação ao número de <i>timesteps</i> (De autoria própria)	65
Figura 27. Matriz de confusão da FCN para o conjunto de teste com 5 <i>timesteps</i> (De autoria própria)	66
Figura 28. Matriz de confusão da FCN para o conjunto de teste com 10 <i>timesteps</i> (De autoria própria)	66
Figura 29. Matriz de confusão da FCN para o conjunto de teste com 15 <i>timesteps</i> (De autoria própria)	67
Figura 30. Matriz de confusão da FCN para o conjunto de teste com 20 <i>timesteps</i> (De autoria própria)	67
Figura 31. Matriz de confusão da FCN para o conjunto de teste com 25 <i>timesteps</i> (De autoria própria)	67
Figura 32. Resultados dos experimentos com a abordagem AC-LSTM-AE utilizando um conjunto de dados com 5 <i>timesteps</i> (De autoria própria).....	70
Figura 33. Resultados dos experimentos com a abordagem OC-LSTM-AE utilizando um conjunto de dados com 5 <i>timesteps</i> (De autoria própria).....	71
Figura 34. Resultados dos experimentos com a abordagem OC-LSTM-AE utilizando um conjunto de dados com 10 <i>timesteps</i> (De autoria própria).....	71
Figura 35. Resultados dos experimentos com a abordagem OC-LSTM-AE utilizando um conjunto de dados com 15 <i>timesteps</i> (De autoria própria).....	72
Figura 36. Resultados dos experimentos com a abordagem OC-LSTM-AE utilizando um conjunto de dados com 20 <i>timesteps</i> (De autoria própria).....	72
Figura 37. Resultados dos experimentos com a abordagem OC-LSTM-AE utilizando um conjunto de dados com 25 <i>timesteps</i> (De autoria própria).....	73

LISTA DE TABELAS

Tabela 1. Estados Operacionais	56
Tabela 2. Variáveis de Estado	57
Tabela 3. Formatação e Características dos Conjuntos de Dados	61
Tabela 4. Características e Distribuição dos Conjuntos de Dados em Treinamento, Validação e Teste.....	62
Tabela 5. Exatidão Alcançada pelas RNPs no Conjunto de Teste	64
Tabela 6. Comparação do Resultado de Diferentes Trabalho na Tarefa de Classificação de Acidentes Nucleares	68
Tabela 7. Comparação do Resultado de Diferentes Trabalho na Tarefa de Gerar Resposta “Não Sei”	73

ACRÔNIMOS E ABREVIACÕES

AE – *AutoEncoder*;

AC-LSTM-AE – *All-Classes Long Short-Term Memory -Autoencoder*;

CAN – *Classificação de Acidentes Nucleares*;

CST – *Classificação de Séries Temporais*;

CNN – *Convolutional Neural Network*;

CUDA – *Compute Unified Device Architecture*;

DA – *Data Augmentation*;

DAE – *Deep Autoencoder*;

DBN – *Deep Belief Network*;

DFNN – *Deep Feedforward Neural Network*;

DRNN – *Deep Rectifier Neural Network*;

EFH – *Engenharia de Fatores Humanos*;

FNN – *Feedforward Neural Network*;

FCN – *Fully Convolution Network*;

GAP – *Global Average pooling Layer*;

GPU – *Graphics Processing Unit*;

IA – *Inteligência Artificial*;

LSTM – *Long Short-Term Memory*;

LSTM-AE – *Long Shot-Term Memory-Autoencoder*;

LSTM-FCN – *Long Shot-Term Memory-Fully Convolution Network*;

MLSTM-FCN – *Multivariate Long Shot-Term Memory-Fully Convolution Network*;

MCAP – *Módulo Classificação de Acidentes Postulados*;

MDA – *Modulo de Detecção de Anomalias*;

MV – *Modulo de Validação*;

MLP – *Multilayer Perceptron*;

NRC – *Nuclear Regulatory Commission*;

OC-NN – *One Class Neural Network*;

OC-LSTM-AE – *One-Class Long Short-Term Memory -Autoencoder*;

PWR – *Pressurized Water Reactor*;

PIDAN – *Problema de Identificação e Diagnóstico de Acidentes Nucleares*;

ReLU – *Rectified Linear Unit*;
RNA – Rede Neural Artificial;
RNR – Rede Neural Rasa;
RNN – *Recurrent Neural Network*;
RNP – Rede Neural Profunda;
SPDS – *Safety Parameter Display Systems*;
SE – *Squeeze-and-Excitation*;
SW – *Sliding Window*;
TMI – *Three Mile Island*

CAPÍTULO 1

INTRODUÇÃO

1.1. APRESENTAÇÃO DO PROBLEMA

Uma usina nuclear é um sistema de alta complexidade, formado por uma grande gama de subsistemas e componentes que são projetados, construídos e operados seguindo normas e diretrizes rigorosas de segurança. O alto padrão de segurança de uma usina nuclear visa garantir que apesar do grande perigo representado pelo inventário de radioatividade no núcleo do reator, a possibilidade de um acidente radiológico seja mínima.

Neste contexto, uma usina nuclear, durante sua operação em condições de normalidade, é constantemente monitorada e controlada por uma equipe de operadores altamente treinada e qualificada. De modo que na ocorrência de um evento anômalo durante a operação, os operadores são responsáveis por identificar e diagnosticar corretamente o evento que está acontecendo, para assim iniciar as ações corretivas apropriadas, garantindo a integridade e segurança da usina nuclear.

A tarefa de identificação e diagnóstico de eventos anômalos em usinas nucleares é considerada um problema de importância crucial, possuindo um alto nível de dificuldade e complexidade associado a si, uma vez que, durante um evento anômalo, os operadores precisam analisar uma grande quantidade de informações originadas dos diversos instrumentos de monitoração e alarmes da usina (JEONG *et al.*, 1996). De forma que, embora a equipe de operadores seja intensamente treinada, dado simplesmente ao elevado número de variáveis em constante evolução, que precisam ser constantemente monitoradas durante o evento, em um curto espaço de tempo, a atuação dos operadores pode ser prejudicada, havendo a possibilidade de ocorrer falhas no diagnóstico.

O diagnóstico incorreto de um evento anômalo pode gerar graves consequências para a integridade da usina nuclear, já que existe a possibilidade que as ações corretivas subsequentes ao diagnóstico falho provoquem a degradação das condições de segurança da usina. Em virtude disso, um simples transiente operacional pode se tornar um acidente de grandes proporções. Um exemplo de tal cenário, é o acidente ocorrido nos Estados Unidos, na usina nuclear de *Three Mile Island* (TMI) em 1979 (BROUGHTON

et al., 1989). O acidente de TMI começou devido a problemas mecânicos, contudo, a situação foi agravada devido a erros cometidos pela equipe de operadores na sala de controle em resposta aos problemas mecânicos iniciais.

Considerando as possíveis consequências severas do problema de identificação e diagnóstico de acidentes nucleares (PIDAN), o campo da Engenharia de Fatores Humanos (EFH), principalmente após TMI, vem dando grande atenção a este tipo de problema. Logo depois do acidente de TMI em 1979, a Comissão de Regulamentação Nuclear (do inglês, *Nuclear Regulatory Commission* – NRC) expediu uma série de documentos normativos que exigiam que diversas modificações críticas fossem realizadas nos projetos das usinas nucleares (LEWIS *et al.*, 1979).

Uma das principais mudanças incentivadas pela NRC foi em relação à interfaces homem-máquina, com a elaboração e implantação de sistemas computacionais de suporte a operação e tomada de decisões, denominados *Safety Parameter Display Systems* (SPDS), na sala de controle. Os SPDS possuem a função de operar como um auxiliar de detecção na sala de controle, com capacidade de processar e organizar rapidamente uma série de informações de operação da planta, de forma a fornecer aos operadores uma visão geral integrada dos parâmetros de operação mais relevantes (JOYCE & LAPINSKY, 1983).

Atualmente, um dos principais desafios da área de EFH, é o desenvolvimento de sistemas capazes de identificar e diagnosticar de forma autônoma um evento anômalo, como um acidente ou transiente, em uma usina nuclear. Posto que tal sistema precisa ser rápido, eficiente e seguro, isso significa que precisa ter capacidade de detectar, em pouco tempo, que um evento anômalo desconhecido está ocorrendo, e a partir disso, classificar corretamente tal evento como um dos acidentes postulados no projeto da usina. Além disso, no aspecto segurança, uma característica imprescindível deste tipo de sistema, é a capacidade de gerar a resposta “Não Sei” para acidentes não postulados, pois, durante um evento anômalo, uma informação incorreta fornecida para a equipe dos operadores pode ser pior que uma situação indeterminada da planta.

Ao longo dos anos, o PIDAN vêm sendo vastamente estudado na literatura científica (BARTLETT & UHRIG, 1992; ALVES, 1993; PEREIRA *et al.*, 1998; KWON & KIM, 1999; MOL *et al.*, 2003; ROCCO & ZIO, 2007; NICOLAU & SCHIRRU, 2017; LEE *et al.*, 2018; PINHEIRO & SCHIRRU, 2019), como um

problema de reconhecimento de padrões, uma vez que durante um evento anômalo é possível observar a evolução no tempo do sinal de saída gerado pelos diversos instrumentos de monitoramento da usina nuclear. Cada um desses instrumentos representa uma variável de estado no PIDAN, considerando que para cada acidente a evolução temporal dessas variáveis gera um padrão específico que caracteriza individualmente os acidentes. Isso faz com que as variáveis de estado sejam peças fundamentais para a identificação e diagnóstico dos eventos anômalos em uma usina nuclear.

Considerando a natureza de reconhecimento de padrões que o PIDAN intrinsecamente possui, através dos anos, diversas abordagens baseadas em técnicas de Inteligência Artificial (IA), vêm sendo exploradas, sozinhas ou associadas a outras técnicas de estatística, no desenvolvimento de sistemas e algoritmos capazes de auxiliar os operadores na tarefa de tomada de decisão, na ocorrência de um evento anômalo, durante a operação de uma usina nuclear. Algumas técnicas de IA que vêm sendo estudadas e avaliadas como possíveis soluções para o PIDAN são: modelo oculto de Markov (do inglês, *hidden Markov model*) (KWON & KIM, 1999); lógica nebulosa (do inglês, *Fuzzy Logic*) (LEE *et al.*, 2018); redes neurais artificiais (RNAs) (BARTLETT & UHRIG, 1992; ALVES, 1993; MOL *et al.*, 2003); máquina de vetores de suporte (do inglês, *support vector machine*) (ROCCO & ZIO, 2007); algoritmos evolucionários (NICOLAU & SCHIRRU, 2017) e algoritmos genéticos (PEREIRA *et al.*, 1998; PINHEIRO & SCHIRRU, 2019).

Dentre as técnicas de IA citadas para resolver o PIDAN, as RNAs tem sido uma das mais exploradas, uma vez que, esta é uma das técnicas mais eficiente para resolver problemas complexos, não lineares, de reconhecimento de padrões. De modo que, é possível afirmar que a evolução na capacidade das RNAs em resolver o PIDAN está acompanhando o aperfeiçoamento das próprias arquiteturas de RNAs.

Tendo em vista que, os primeiros trabalhos que utilizavam RNAs para resolver PIDAN (BARTLETT & UHRIG, 1992; ALVES, 1993) embora tivessem potencial, ainda não alcançavam um nível de desempenho para serem consideradas uma solução viável para o PIDAN, principalmente quando o cenário proposto era composto por múltiplos acidentes para serem classificados. Contudo, com o avanço das arquiteturas, em pouco tempo, as RNAs evoluíram ao ponto de lidar com cenários compostos por diversos acidentes (MOL *et al.*, 2003). Nesse contexto, o advento, na última década, do

novo paradigma de RNAs nomeado aprendizado profundo (do inglês, *deep learning*), que se caracteriza pela utilização de arquiteturas de redes neurais profundas (RNPs) formadas por múltiplas camadas interconectadas, compostas por centenas até milhares de neurônios, está aperfeiçoando ainda mais a capacidade das RNAs na resolução do PIDAN.

Nos últimos anos, o uso de RNPs tem aumentado significativamente em diversas áreas e aplicações, devido ao alto desempenho que as RNPs estão proporcionando na resolução de problemas complexos e na melhoria dos resultados em várias tarefas. Atualmente, existem sistemas baseados em RNPs capazes de realizar tarefas de: processamento de linguagem natural de voz e texto; reconhecimento visual; recomendações e suporte a decisões; entre outras (SANTHANAM *et al.*, 2019). No entanto, na área da engenharia nuclear, ainda são relativamente poucos os estudos que exploram o uso das RNPs para resolver os problemas e desafios dessa área. Contudo, recentes trabalhos (PENG *et al.*, 2018; SANTOS *et al.*, 2019a; SANTOS *et al.*, 2019b; PINHEIRO *et al.*, 2020; SAEED *et al.*, 2020; YANG *et al.*, 2020; SANTOS *et al.*, 2021), já revelam que os modelos contemporâneos de RNPs, quando comparadas com os modelos clássicos de RNAs e com outras técnicas clássicas de IA, apresentam um desempenho superior na capacidade de resolver problemas complexos, como o PIDAN, conseguindo resultados mais exatos, com um menor tempo de processamento computacional.

1.2. OBJETIVOS

Como já mencionado, a temática do desenvolvimento de um sistema de identificação e diagnóstico de acidentes em usinas nucleares, através dos anos, vem sendo vastamente estudada. Contudo, esses tipos de sistema ainda não foram realmente implantados em usinas nucleares, dados os requisitos de confiabilidade, exatidão e velocidade de processamento que tal sistema precisa possuir para que realmente seja viável e segura a sua implantação.

Neste contexto, é possível afirmar que um sistema para o PIDAN tem que conseguir resolver duas tarefas principais, a de classificação de séries temporais e a de detecção de novidades. Discutindo a primeira tarefa, a de classificação de séries temporais (CST), está relacionada com o problema principal que o sistema se propõe a resolver, a classificação de acidentes nucleares (CAN). Uma vez que, no problema de CAN, cada acidente é representado pelo comportamento temporal de diversas variáveis

de estado da usina, ou seja, para cada acidente é definido um conjunto de séries temporais que definem este acidente. Deste modo, a resolução desse problema é encontrada por meio de métodos capazes de classificar corretamente os acidentes, a partir das séries temporais que descrevem o comportamento destes acidentes. Inicialmente, alcançar um elevado nível de exatidão na CAN era extremamente difícil. Contudo, atualmente, com os modelos contemporâneos de RNP, está sendo possível atingir altos níveis de exatidão na CAN, de tal forma, que artigos publicados já demonstram que modelos de RNP conseguem alcançar exatidão de 99% na CAN (SANTOS *et al.*, 2019a; SANTOS *et al.*, 2019b; PINHEIRO *et al.*, 2020; SANTOS *et al.*, 2021).

Embora através do uso de RNPs a eficiência dos sistemas na resolução da tarefa de CAN tenha evoluído de forma notável, ainda não é suficiente para garantir a viabilidade e segurança do sistema. Pois, como citado, esse tipo de sistema também tem que conseguir solucionar a tarefa de detecção de novidades. Esta tarefa se caracteriza como o problema de identificar padrões de dados novos, desconhecidos ou não observados, pelo modelo classificador. No contexto de um sistema para o PIDAN, a capacidade de detectar novidades, significa fornecer ao sistema a habilidade de gerar a resposta “Não Sei”. Assim como, para a CAN, abordagens utilizando RNPs começaram a ser propostas para a tarefa de detecção de novidades, mostrando resultados promissores (PINHEIRO *et al.*, 2020).

Apesar dos resultados promissores que RNPs conseguem alcançar nas tarefas principais de um sistema para o PIDAN, um dos problemas fundamentais desse tipo de sistema permanece: a estruturação do sistema de forma a alcançar um alto nível de exatidão (próximo dos 99%), tanto quando confrontado com dados das classes conhecidas, quanto quando confrontado com dados desconhecidos (nesse caso gerando resposta “Não Sei”). Pois, o que geralmente acontece é que, quando o sistema consegue atingir um alto nível de exatidão na identificação dados conhecidos, o seu nível de detecção de dados desconhecidos cai. Em contrapartida, quando o sistema consegue atingir um alto nível de exatidão na detecção de dados desconhecidos, o seu nível de identificação de dados conhecidos cai. Tal problema ainda é uma das barreiras que inviabilizam a implantação real desse tipo de sistema.

Sendo assim, visando desenvolver um sistema capaz de alcançar um elevado nível de desempenho tanto na tarefa de classificação de acidentes postulados, quanto na

de geração de resposta “Não Sei”, neste trabalho é proposto o desenvolvimento de um sistema com estrutura modular, baseado em arquiteturas profundas de redes neurais, para classificação de acidentes postulados em usinas nucleares com capacidade de identificação de anomalia e resposta “Não Sei”.

Em relação aos diferenciais do sistema proposto, a estrutura modular permitirá que módulos específicos sejam confeccionados para a realização de cada tarefa de um sistema para o PIDAN, de modo que, ao utilizar módulos individuais para cada tarefa, cujo núcleo principal do funcionamento será um ou mais modelos de RNP, evitará que, por exemplo, um alto nível de exatidão na identificação de dados de classes conhecidas, influencie negativamente a detecção de dados desconhecidos. Pois os módulos só se comunicarão entre si para informar a sua saída, de modo que o modo de operação individual de um módulo não influenciará diretamente os outros.

Sendo assim, para cada um dos módulos será analisado o desempenho de diferentes arquiteturas de RNPs, incluindo algumas arquiteturas consideradas estado da arte para o problema de classificação de séries temporais, com o objetivo de encontrar aquela que possui melhor capacidade para executar a tarefa requisitada pelo módulo.

Ademais, uma característica que distingue esse estudo é a complexidade, em questão de número de eventos contemplados e similaridade entre eles, do conjunto de dados que será utilizado para validar o desempenho dos módulos do sistema proposto. Posto que o conjunto de dados que será utilizado é formado por 15 acidentes postulados mais a operação normal para a usina nuclear Angra 2. Dentre esses 15 acidentes, alguns possuem comportamento inicial extremamente parecido, de maneira que, alcançar um nível de exatidão aceitável na classificação destes eventos específicos é um desafio que ainda não foi superado, e que este trabalho se propõe a superar.

Portanto, este trabalho apresentado compõe mais uma etapa da linha de estudos construída pelo Laboratório de Monitoração de Processo (LMP) da COPPE (ALVES, 1993; MOL *et al.*, 2003; SANTOS *et al.*, 2019a; SANTOS *et al.*, 2019b; PINHEIRO *et al.*, 2020; SANTOS *et al.*, 2021), que vem amadurecendo através do tempo, focada no desenvolvimento de um sistema para o PIDAN baseado em RNAs. O objetivo final desse trabalho, é fazer com que o sistema modular proposto alcance um elevado nível de confiabilidade, eficiência e velocidade de processamento computacional.

CAPÍTULO 2

FUNDAMENTAÇÃO TEÓRICA

2.1. REVISÃO BIBLIOGRÁFICA DAS TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL APLICADAS A DE IDENTIFICAÇÃO E DIAGNÓSTICO DE ACIDENTES NUCLEARES

Uma vez que o PIDAN é um problema complexo de grande relevância na área da engenharia nuclear, na literatura é possível encontrar uma vasta gama de trabalhos que exploram a utilização de técnicas de IA com o objetivo de solucionar o PIDAN. Sendo assim, nessa seção, alguns dos principais estudos que tentam solucionar o PIDAN através de técnicas de IA serão explorados.

2.1.1. Técnicas Clássicas de Inteligência Artificial

(PEREIRA et al., 1998) modelaram um algoritmo genético para classificar 3 acidentes nucleares de um reator de água pressurizada (do inglês, *pressurized water reactor* – PWR), sendo que cada acidente era representado por várias séries temporais em um banco de dados, de modo que a ideia central era encontrar agrupamentos no conjunto dos padrões de treinamento, de cada acidente, para que seus centroides pudessem distinguir os acidentes com um mínimo de erros de classificação. O resultado alcançado por esse método foi uma exatidão média de 95%, em um total de 180 experimentos.

(NICOLAU & SCHIRRU, 2017) desenvolveram um sistema para o PIDAN, com capacidade de classificar um conjunto de acidentes nucleares, e também capaz de gerar resposta “Não Sei” quando confrontado com uma situação desconhecida. O método de classificação foi baseado em um algoritmo evolucionário quântico, sendo este responsável por encontrar o vetor representativo de cada classe de acidente. Já o método para gerar a resposta “Não Sei”, foi baseado na teoria dos vizinhos mais próximos dos diagramas de Voronoi, sendo este responsável por determinar as "áreas de influência", em torno dos vetores representativos encontrados pelo algoritmo evolucionário quântico. Os resultados das simulações realizadas, para acidentes de um reator do tipo PWR, mostraram que o sistema foi capaz de identificar os acidentes de referência e distinguir os tipos desconhecidos de estados operacionais.

(PINHEIRO & SCHIRRU, 2019) apresentaram, pela primeira vez, a aplicação da programação genética ao PIDAN. A metodologia consistiu em avaliar a eficiência da

programação genética, como técnica para otimização e geração de características em um sistema de reconhecimento de padrões, para o diagnóstico de acidentes em uma usina nuclear do tipo PWR. Neste estudo, foram considerados 4 estados operacionais, com a programação genética sendo aplicada a desenvolvimento de classificadores não lineares, capazes de fornecer a maior quantidade de informações discriminatórias para cada um dos estados e, conseqüentemente, melhores taxas de identificação. Como resultado, a programação genética provou ser uma metodologia capaz de atingir taxas de sucesso de 100% na classificação dos 4 estados operacionais considerados.

2.1.2. Arquiteturas Clássicas de Redes Neurais Artificiais

(BARTLETT & UHRIG, 1992) propuseram o uso de uma RNA para classificar sete acidentes de um reator de água fervente (do inglês, Boiling Water Reactor). A arquitetura de RNA utilizada foi uma rede neural progressiva (do inglês, feedforward neural network – FNN), que possuía além das camadas de entrada e saída, apenas uma camada oculta composta por cerca de dez neurônios. Vale ressaltar que este foi um dos primeiros trabalhos a propor o desenvolvimento de um sistema de suporte a tomada de decisão dos operadores, utilizando técnicas de IA, de modo que, a principal contribuição deste trabalho foi mostrar a viabilidade do uso de RNAs para resolver o PIDAN.

(ALVES, 1993) desenvolveu um sistema automático baseado em FNNs para a identificação de 14 acidentes de um reator PWR. O sistema elaborado foi testado através de simulações, sendo capaz de determinar, com exatidão, 60% dos acidentes contemplados. Contudo, o diferencial desse trabalho foi a adição de um módulo ao sistema incumbido de gerar a resposta “Não Sei”. No caso, quando, por algum motivo, um evento não era reconhecido pelo sistema de identificação, um módulo, baseado em um sistema especialista, inquiria os operadores para obter informações adicionais a respeito das condições da usina. Este foi um dos primeiros estudos a atribuir a um sistema para o PIDAN, baseado em RNAs, um modo de lidar com eventos desconhecidos (eventos fora do escopo do treinamento da RNA), ou seja, atribuir ao sistema a capacidade de gerar a resposta “Não Sei”, que é uma capacidade essencial para esse tipo de sistema.

(MOL et al., 2003) elaboraram um sistema baseado em duas redes neurais de “pulo” (do inglês, 'jump' neural network), sendo que a primeira rede era utilizada para classificar 15 acidentes postulados, simulados para um PWR. Já a segunda rede tinha a

função de validar a classificação (da primeira rede) através da validação de cada variável, ou seja, a segunda rede atribuída ao sistema a capacidade de gerar uma resposta “Não Sei” para eventos desconhecidos.

2.1.3. Arquiteturas Profundas de Redes Neurais Artificiais

(PENG *et al.*, 2018) utilizaram um novo método de diagnóstico de acidentes em usinas nucleares baseado em análise de correlação e rede de crença profunda (do inglês, *deep belief network* – DBN). Nesse caso, a análise de correlação foi usada para reduzir o número de variáveis do conjunto de dados de acidentes, e a DBN foi usada para classificar os acidentes. O método proposto foi validado com um conjunto de dados, simulado via PCTTRAN (CHENG *et al.*, 2012), contendo 7 estados operacionais (6 acidentes e a operação normal) e 64 variáveis. Este foi um dos primeiros trabalhos a utilizar uma arquitetura de RNP no PIDAN, e o resultado obtido foi promissor, com a DBN conseguindo classificar os 7 estados operacionais com uma exatidão em volta dos 99%.

(SANTOS *et al.*, 2019a) implementou uma rede neural retificadora profunda (do inglês, *deep rectifier neural network* – DRNN) para classificar 13 estados operacionais (12 acidentes e a operação normal) simuladas para a usina nuclear Angra 2. A abordagem proposta alcançou uma exatidão de aproximadamente 99%, dependendo se o processo de classificação da DRNN fosse iniciado no primeiro ou terceiro segundo após o início do evento anômalo, respectivamente. Esta abordagem foi considerada estado da arte pelos autores, devido ao número de estados operacionais para serem classificados, a alta taxa de exatidão obtida, o tempo razoável de treinamento (dado o *hardware* utilizado), a versatilidade de configuração, parametrização e a fácil capacidade de extensão para mais acidentes.

(SANTOS *et al.*, 2019b) analisaram o uso de múltiplas RNPs do tipo autocodificadora profunda (do inglês, *deep autoencoder* – DAE) para classificar 13 estados operacionais (12 acidentes postulados mais a operação normal) simulados para a usina nuclear Angra 2. Nesta abordagem, cada DAE foi treinado com dados de apenas um dos 13 estados operacionais, ou seja, cada DAE tinha capacidade de identificar apenas um acidente. A premissa dessa abordagem era que para entradas suficientemente semelhantes aos padrões utilizados para o treinamento, o DAE geraria um baixo erro, enquanto para entradas suficientemente diferentes, o erro seria alto.

Sendo assim, seria possível identificar a qual estado operacional uma entrada pertence, por intermédio do DAE que gerou o menor erro para essa entrada. Tal abordagem foi capaz de alcançar uma exatidão de cerca de 96%, sendo um dos primeiros estudos a usar redes do tipo autocodificadora no PIDAN.

(PINHEIRO *et al.*, 2020) continuaram o estudo iniciado nos trabalhos de Santos *et al.* (2019a) e Santos *et al.* (2019b), e desenvolveram novas abordagens para fornecer um sistema para o PIDAN, baseado em aprendizagem profunda, a capacidade de gerar resposta “Não Sei”. Foram avaliadas diversas abordagens usando dados simulados de 13 estados operacionais para um PWR. A abordagem mais bem-sucedida foi uma baseada em DAEs, esta foi capaz de alcançar uma exatidão média de 99% em termos de classificações corretas "Não Sei".

(SAEED *et al.*, 2020) Propuseram um sistema para o PIDAN, com capacidade de gerar resposta “Não Sei”, baseado em análise de componentes principais e RNPs, mais especificamente redes recorrentes e convolucionais. O estudo foi realizado no modelo de simulação da usina nuclear IP-200, usando o código termo-hidráulico RELAP5/MOD 4.0 (PEREZ *et al.*, 2015). O sistema proposto foi verificado utilizando 4 tipos diferentes de acidentes e 1 caso desconhecido. Os resultados alcançados sugerem que o sistema proposto é um esquema viável para o PIDAN.

(YANG *et al.*, 2020) estudaram o desenvolvimento de um sistema para o PIDAN que faz uso de uma arquitetura de RNP do tipo recorrente, mais especificamente uma *long short-term memory* (LSTM), para identificar 5 tipos de acidentes nucleares, além de uma rede autocodificadora para gerar resposta “não sei”. Os acidentes foram simulados usando o simulador nuclear compacto que é baseado em um reator do tipo PWR. Na melhor configuração da LSTM, essa abordagem conseguiu alcançar uma exatidão de aproximadamente 99% na classificação de acidentes.

2.2. CONTEXTO HISTÓRICO DAS REDES NEURAIS ARTIFICIAIS

Segundo Goodfellow *et al.*, (2016) a evolução histórica das pesquisas sobre redes neurais artificiais pode ser dividida em três eras:

- **Era da Cibernética:** Esta primeira era das RNAs se iniciou em 1960, e se estendeu até o final da década de 1980. Esta era se caracterizou pelo surgimento das primeiras teorias sobre o funcionamento do neurônio biológico e a implantação dos primeiros modelos de neurônio artificial.

- **Era do Conexionismo:** A segunda era é definida pela introdução do algoritmo de retropropagação, que permitia que redes neurais artificiais com mais de uma camada oculta fossem treinadas.
- **Era do Aprendizado Profundo:** A era contemporânea das RNAs, é caracterizado por viabilizar o desenvolvimento e treinamento, rápido e eficiente, de redes neurais que agora podem ser compostas de dezenas de camadas ocultas e milhares de neurônios artificiais.

No restante deste capítulo, de forma breve, serão apresentadas as pesquisas que caracterizaram cada uma destas eras.

2.2.1. Era da Cibernética

As bases computacionais para a pesquisa e desenvolvimento de RNAs começaram a ser erguidas no início dos anos 40, por dois pesquisadores, o neurofisiologista Warren Sturgis McCulloch e o matemático Walter Pitts.

McCulloch e Pitts foram inspirados por estudos sobre o funcionamento do neurônio biológico, principalmente pelos mecanismos de ativação dos neurônios, que pode ser descrito de forma simples: se a entrada (impulso elétrico) for forte o suficiente, o neurônio dispara, ou seja, transmite esse impulso para os demais neurônios que estão conectados consigo; caso contrário, o neurônio permanece no nível básico e não transmite o impulso. Partindo deste princípio, o neurônio pode, abstratamente, ser considerado um elemento de computação binária com uma saída de 1 (disparo) ou 0 (repouso). Assim, máquinas lógicas (booleanas) poderiam ser construídas juntando esses neurônios binários simples com o objetivo de mostrar uma maneira pela qual uma linguagem lógica de pensamento poderia surgir de um sistema nervoso (SHARKEY et al., 2000).

A partir desta premissa, McCulloch & Pitts, (1943) modelaram o primeiro esboço de uma RNA, chamado modelo neural MCP (McCulloch–Pitts), este pode ser considerado o modelo ancestral das RNAs. O MCP utilizava circuitos eletrônicos e, o neurônio artificial concebido para este modelo era capaz de receber entradas binárias, ou seja, valores lógicos 0 ou 1, que representavam o estado do neurônio artificial “desativado” ou “ativado”, advindas de outros neurônios, e caso o valor de entrada recebido fosse maior que um limiar predefinido, gerava uma saída também binária.

Embora o modelo MCP tenha alcançado sucesso, conseguindo mapear algumas funções lógicas lineares, este ainda não possuía a capacidade de aprendizado. Em virtude disso, inspirado pelos estudos teóricos conduzidos pelo psicólogo Donald O. Hebb sobre o funcionamento do processo de aprendizagem neurológico, o também psicólogo Frank Rosenblatt construiu um modelo de neurônio artificial capaz de aprender, chamado perceptron.

Nesse contexto, segundo Hebb (1949), no cérebro, o aprendizado ocorre principalmente através da formação e fortalecimento das conexões (sinapses) entre neurônios biológicos específicos. Seguindo esta premissa, o perceptron, proposto por Rosenblatt (1958), introduziu o conceito dos pesos sinápticos, tradicionalmente chamados somente de pesos, que representavam a força que cada entrada de um neurônio deveria ter para gerar a saída desejada, de modo que, o aprendizado acontecia através do ajuste desses pesos.

Um perceptron é, fundamentalmente, uma função linear dos sinais de entrada e em virtude disso, sua capacidade de aprendizado era limitada somente a funções lineares, operações lógicas como NOT, AND ou OR. Tal limitação foi levantada por Minsky & Papert (1969) que mostraram que o perceptron não podia resolver funções não lineares, como XOR ou NXOR (WANG & RAJ, 2017). Além disso, Minsky & Papert (1969) argumentaram que para que o perceptron fosse capaz de lidar com funções não lineares seriam necessárias múltiplas camadas de perceptrons, contudo, o modelo de aprendizado do perceptron de Rosenblatt não funcionava para múltiplas camadas. Como resultado, até os anos 80, os estudos envolvendo RNAs foram de certa forma abandonados, com poucas pesquisas sendo realizadas.

2.2.2. Era do Conexionismo

A era do conexionismo foi guiada pela ideia de que se um grande número de unidades de processamento, como os neurônios artificiais, trabalhasse de forma interconectada em uma rede, assim como acontece com neurônios biológicos no sistema nervoso, seria possível fazer com que esses modelos computacionais atingissem um comportamento inteligente. Seguindo esta premissa, foi nesta era que surgiram diversos conceitos essenciais, que na época permitiram a evolução das RNAs, e que até hoje servem de base para o desenvolvimento desses modelos.

O principal conceito que definiu a era conexionista foi a implantação eficiente do algoritmo de retropropagação de erros para ajuste de pesos. Embora este algoritmo tenha sido proposto anteriormente por Werbos, (1974), foi somente através do trabalho de *Rumelhart et al.*, (1986) que este se popularizou. A partir do trabalho *Rumelhart et al.*, (1986) se tornou possível o treinamento eficiente de RNAs compostas por múltiplas camadas de perceptron, as quais foram nomeadas como redes neurais perceptron de múltiplas camadas (do inglês, *multilayer perceptrons* – MLP).

Outro conceito introduzido também nesta época foi a representação distribuída (*HINTON et al.*, 1986), este introduz a ideia de que cada entrada de um sistema deve ser representada por diversas características, assim como cada característica deve representar o maior número possível de entradas. O conceito de representação distribuída é melhor compreendido ao se utilizar como exemplo um sistema que pode reconhecer objetos como carros, caminhões e pássaros, sendo que cada um desses objetos pode ter 3 cores, vermelho, verde ou azul. Sendo assim, uma forma de representar as entradas desse sistema seria ter um neurônio para cada uma das nove combinações possíveis: caminhão vermelho, carro vermelho, pássaro vermelho, caminhão verde, etc. Ou seja, o sistema teria 9 neurônios diferentes, onde cada neurônio deveria aprender independentemente o conceito de cor e o tipo do objeto. Outra forma de organizar esse sistema seria através de uma representação distribuída, com três neurônios descrevendo a cor do objeto e três neurônios descrevendo o tipo de objeto. Neste tipo de representação apenas 6 neurônios seriam necessários, e a principal vantagem é que, por exemplo, o neurônio que descreve a vermelhidão seria capaz de aprender sobre a vermelhidão através das imagens de carros, caminhões e pássaros, e não apenas por imagens de uma categoria específica de objeto.

Ademais, durante o conexionismo, houve também avanços significativos no estudo de RNAs para modelagem de imagens e de sequências. Em relação ao processamento de imagens, Fukushima (1980) introduziu um modelo de arquitetura de RNA, chamada *Neocognitron*, inspirado na estrutura do sistema visual dos mamíferos, para processamento de imagens. A *Neocognitron* foi posteriormente utilizada como base para o desenvolvimento da RNA de arquitetura convolucional (*LECUN*, 1989), que é uma das principais arquiteturas de RNA da atualidade.

Já no campo da modelagem de sequência com RNAs, *Hochreiter*, (1991) e *Bengio et al.*, (1994) identificaram e definiram o problema de desaparecimento de

gradiente que não permitia que as redes neurais recorrentes, que são as arquiteturas de RNA especializadas no processamento de sequências, conseguissem modelar sequências longas. Posteriormente, Hochreiter e Schmidhuber, (1997) introduziram a *Long Short Term Memory* (LSTM), que é uma RNA de arquitetura recorrente, capaz de modelar sequências longas. Atualmente, a LSTM é uma das principais RNAs para a modelagem de sequência.

O problema de desaparecimento de gradiente, não só afligia as redes recorrentes, como também afetava as MLPs. As MLPs com muitas camadas ocultas, geralmente com mais de 4 camadas ocultas, não conseguiam ser treinadas de forma eficiente através do algoritmo de retropropagação, pois o gradiente, que é utilizado para ajustar os pesos da rede em relação ao erro obtido na saída, diminuía ao ser retropropagado pelas camadas da rede, de modo que, em uma RNA com muitas camadas ocultas, o gradiente ao chegar às primeiras camadas da rede era tão pequeno que o ajuste dos pesos dessas camadas se tornava inviável. Em virtude disso, por muito tempo, o uso de RNAs com muitas camadas ocultas era inviável, ou seja, a capacidade de aprendizado que as RNAs desta época podiam alcançar era limitado.

O encerramento da era do conexionismo ocorreu no final da década de 90, a partir de uma forte redução no interesse em pesquisas envolvendo RNAs. Um dos motivos que levou à redução de interesse, foi que no início do conexionismo, muitas expectativas ambiciosas e irrealistas sobre a capacidade das RNAs foram divulgadas como uma forma de atrair investimentos para pesquisas na área, contudo, quando as RNAs não conseguiram entregar as expectativas prometidas, muito devido ao problema de desaparecimento de gradiente, os investidores ficaram desapontados e desistiram do investimento em pesquisas que envolviam RNAs. Além disso, simultaneamente, outras técnicas de inteligência artificial começaram a alcançar bons resultados em diversas tarefas importantes, fazendo com que as RNAs ficassem ainda mais ofuscadas.

2.2.3. Era do Aprendizado Profundo

A era do aprendizado profundo, que é o período contemporâneo das RNAs, se caracterizou pela superação do problema de desaparecimento de gradiente que limitava o treinamento de RNAs com múltiplas camadas ocultas, que neste período ganharam o nome de redes neurais profundas. Sendo assim, a era do aprendizado profundo foi iniciada oficialmente em 2006, com o trabalho de Hinton *et al* (2006) que criou a

arquitetura de RNP conhecida como DBN, e utilizando uma estratégia de pré-treinamento conseguiu fazer com que uma RNP fosse treinada de maneira eficiente. A partir deste trabalho, o interesse em RNAs retornou e uma nova onda de trabalhos sobre o assunto voltou a ocorrer.

A primeira leva de trabalhos desta era focaram, principalmente, no aprofundamento do estudo teórico sobre como o aprendizado acontecia nas RNPs e no desenvolvimento de arquiteturas de RNPs que pudessem ser treinadas de maneira eficiente. Além disso, foi a partir desses estudos que o uso do termo "aprendizagem profunda" se popularizou, pois este termo era utilizado para enfatizar que os modelos de RNAs agora possuíam múltiplas camadas ocultas. Atualmente, aprendizado profundo acabou se tornando o nome do campo de estudo que envolve RNAs, assim como muitas vezes o termo aprendizado profundo é utilizado como sinônimo para as próprias RNAs, de maneira que as redes profundas utilizadas atualmente podem ser chamadas também de modelos de aprendizado profundo.

Embora inicialmente a estratégia de pré-treinamento tenha alcançado sucesso no treinamento de RNPs, foi somente em 2010 que começaram a surgir arquiteturas de RNPs que conseguiam ser treinadas de forma eficiente sem estratégias de pré-treinamento. De modo que Glorot e Bengio, (2010) ao realizar um estudo aprofundado sobre o treinamento de RNPs, mostraram que o principal motivo do problema de desaparecimento de gradiente em RNPs é o uso de funções de ativação do tipo sigmoide. Em virtude disso, Glorot *et al* 2011), então propuseram um novo tipo de arquitetura RNP, chamada de rede neural profunda retificada (do inglês, *Deep Rectifier Neural Network* - DRNN), que para ativação dos seus neurônios internos, substituíam as funções sigmoide por uma função retificada, mais especificamente a *rectified linear unit* (ReLU), de modo que a DRNN não sofre do problema de desaparecimento de gradiente.

Rapidamente o uso de funções retificadas em RNPs se popularizou e, em 2012, o trabalho de Krizhevsky *et al.*, (2012) consolidou e impulsionou a era do aprendizado profundo. Krizhevsky *et al.*, (2012), desenvolveram uma RNP de arquitetura convolucional, cujos neurônios internos eram ativados por ReLu, chamada AlexNet, que foi capaz de vencer o maior e mais difícil desafio da área de reconhecimento de objetos em imagens: tal desafio é chamado *Imagenet Large Scale Visual Recognition Challenge* (ILSVRC) (RUSSAKOVSKY *et al.*, 2015). Na edição 2012 do ILSVRC, a AlexNet alcançou um resultado do nível do estado da arte, superando todas as demais

abordagens clássicas de inteligência artificial e visão computacional. O resultado obtido pela AlexNet significou um avanço significativo no campo da visão computacional, e a partir deste ponto na história, o interesse pelo aprendizado profundo aumentou rapidamente.

Desde 2012, a área do aprendizado profundo continua a crescer rapidamente, com RNPs alcançando resultados do nível do estado da arte em tarefas como: processamento de linguagem natural (BROWN *et al.*, 2020), processamento de imagens (MOEN *et al.*, 2019), previsão de séries temporais (QIU *et al.*, 2014), detecção de anomalia (DU *et al.*, 2017), descoberta de drogas (MAYR *et al.*, 2016), identificação de câncer (BEJNORDI *et al.*, 2017; ESTEVA *et al.*, 2017; COUDRAY *et al.*, 2018; WANG *et al.*, 2016), entre muitas outras tarefas. Atualmente, várias aplicações de diversas indústrias implementam RNPs e são altamente lucrativas, de modo que o aprendizado profundo agora é usado por muitas empresas de tecnologia de ponta, incluindo Google (WU *et al.*, 2016; POWLES & HODSON, 2017), Microsoft (DENG *et al.*, 2013), Facebook (PARK *et al.*, 2018), Baidu (HANNUN *et al.*, 2014).

2.3. PRINCIPAIS ARQUITETURAS DE REDES NEURAIAS

2.3.1. Redes Neurais Perceptron Multicamadas

A rede neural perceptron multicamadas (do inglês, *multilayer perceptron* - MLP), atualmente também chamada de rede neural progressiva totalmente conectada (do inglês, *fully connected feedforward neural network*), é a arquitetura de rede neural mais popular, sendo que, muitas vezes a imagem da arquitetura da MLP, como mostrada na Figura 1, é utilizada para representar o que é uma RNA.

A MLP, representada pela Figura 1, possui no mínimo três camadas: a camada de entrada que recebe os dados de entrada; a camada oculta que é composta por neurônios artificiais ativados por uma função não linear, sendo responsável por aprender a representação do problema a ser modelado, e a camada de saída que gera a saída desejada. Ademais, na MLP, todos os neurônios de uma camada se conectam, através de pesos sinápticos (parâmetros da rede), com todos os neurônios da camada anterior e posterior. Em virtude disso, essa é considerada uma arquitetura totalmente conectada. Além disso, a MLP também é considerada arquitetura progressiva, uma vez que a informação flui em apenas uma direção, da camada de entrada para a camada de saída, não havendo

nenhum tipo de recorrência (GOODFELLOW *et al.*, 2016; RASCHKA & MIRJALILI, 2017).

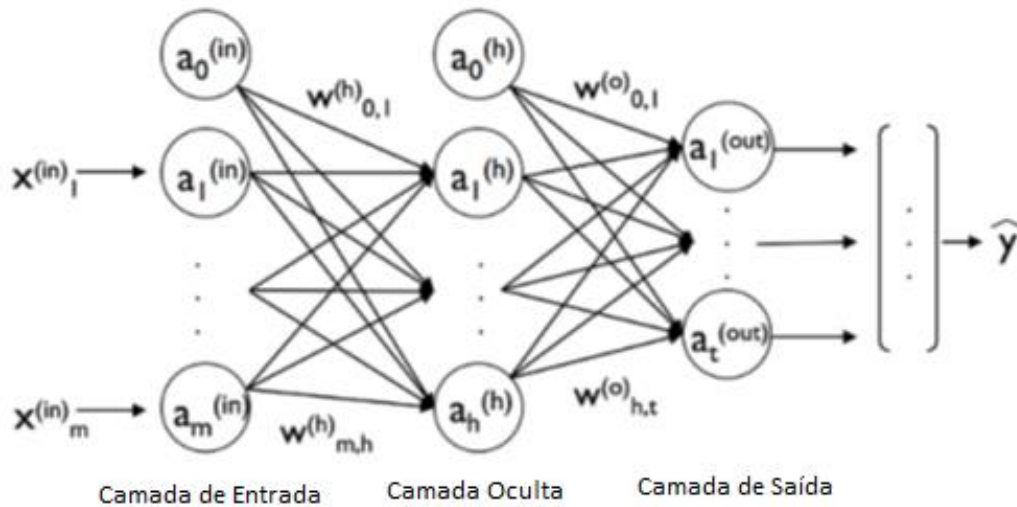


Figura 1. Estruturas básica de uma MLP (RASCHKA & MIRJALILI, 2017)

O objetivo de uma MLP, é aprender por meio de suas camadas ocultas a função f capaz de mapear os dados de entrada para a saída desejada, por exemplo, em um problema de classificação, que pode ser definido como $y = f(x)$, significando que a função f , mapeia a entrada x para categoria y . Nesse caso, uma MLP pode ser representada por um mapeamento $y = f(x; \theta)$, onde θ são os valores dos parâmetros de aprendizado da rede que permitem que a função f mapeie a entrada x para a saída y (GOODFELLOW *et al.*, 2016).

Como citado, a MLP tem, no mínimo, uma camada oculta, contudo, quando a MLP tem mais de uma camada oculta, ela também pode ser chamada de rede neural progressiva profunda (do inglês, *deep feedforward neural network* - DFNN). A Figura 2 destaca a diferença entre uma MLP simples e uma DFNN. Na construção de uma MLP, o número de camadas ocultas e de neurônios são hiperparâmetros que são definidos pelo engenheiro de forma empírica, dependendo do tipo do problema a ser modelado e a quantidade de dados disponíveis para treinamento.

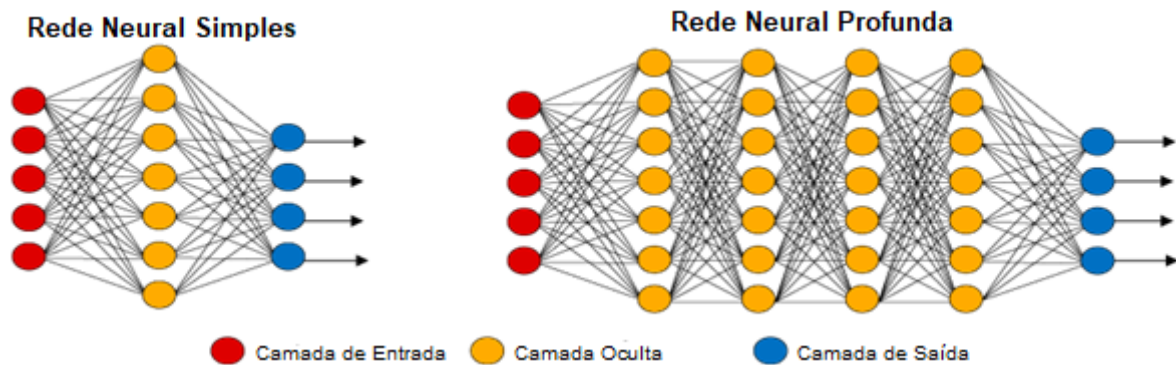


Figura 2. Esquerda: Uma MLP simples com uma camada oculta. Direita: Uma DFNN com 4 camadas ocultas (VÁZQUEZ, 2017)

Neste contexto, a escolha do número de camadas ocultas é um aspecto central para definir a capacidade da MLP, assim como de outras arquiteturas de RNA, pois, há uma diferença substancial na capacidade de aprendizagem de RNAs com apenas uma camada oculta, atualmente nomeadas redes neurais rasas (RNR) e as RNAs com múltiplas camadas, as redes neurais profundas (RNPs). A principal vantagem das RNPs é que o uso de múltiplas camadas ocultas permite que o aprendizado aconteça de forma hierárquica, seguindo uma abordagem de aprendizado chamada de hierarquia de representações de características. A premissa dessa abordagem é que o aprendizado de conceitos complicados acontece a partir do aprendizado de outros mais simples (GOODFELLOW *et al.*, 2016), de forma que, no processo de aprendizado das RNPs, as camadas mais profundas da rede aprendem as características complexas do problema que está sendo modelado, com base nas características simples aprendidas pelas primeiras camadas. Portanto, as RNPs conseguem representar e aprender funções complexas com muito mais eficiência que as RNR (LECUN *et al.*, 2015; SCHMIDHUBER, 2015; SRIVASTAVA *et al.*, 2015; BENUWA *et al.*, 2016).

Embora as DFNNs apresentem um potencial de aprendizado muito mais elevado que as MLPs clássicas (com uma camada oculta), o treinamento de DFNNs é mais complexo, pois há um número muito mais elevado de transformações parametrizadas que um sinal de entrada encontra conforme se propaga da camada de entrada para a de saída, de modo que, por muito tempo a construção de DFNNs com mais de 4 camadas ocultas foi limitada devido a problemas no treinamento, como o desaparecimento de gradiente.

2.3.1.1. O Problema de Desaparecimento de Gradiente

O desaparecimento de gradiente é um problema que afeta RNPs treinadas via algoritmos clássicos de retropropagação de gradiente. Uma vez que, em RNAs, o aprendizado acontece através do ajuste dos parâmetros da rede, e para ajustar esses parâmetros, o gradiente do erro em relação a cada parâmetro é utilizado. Esses gradientes são encontrados através do algoritmo de retropropagação, que usando a regra da cadeia, calcula camada a camada, indo da camada de saída em direção à entrada, o gradiente dos parâmetros das camadas ocultas em relação ao erro. Contudo, quando essa abordagem é aplicada em RNPs, que possuem múltiplas camadas ocultas, acontece o problema em que o valor do gradiente vai diminuindo conforme o algoritmo avança de camada para camada, de modo que o valor do gradiente se torna extremamente pequeno ao chegar nas primeiras camadas ocultas e, com isso, o ajuste dos parâmetros dessas camadas é praticamente inexistente, fazendo com que o treinamento se torne lento e nunca convirja para um bom resultado.

Embora o desaparecimento de gradiente já fosse conhecido e observado empiricamente desde a era conexionista das RNAs, foi só a partir do trabalho de Glorot e Bengio, (2010) que foi determinado que o principal fator contribuindo para o problema de desaparecimento de gradiente era o uso de funções sigmoide logísticas, que eram as mais populares na época, para ativar os neurônios ocultos das RNPs. Por exemplo, a função sigmoide logística, que é definida como $f(x) = \frac{1}{1+e^{-x}}$, possui limites assintóticos entre 0 e 1, conforme suas entradas se aproximam de 4 ou -4, como pode ser observado na Figura 3. Nessas regiões, suas derivadas se aproximam de zero e o gradiente tende a desaparecer.

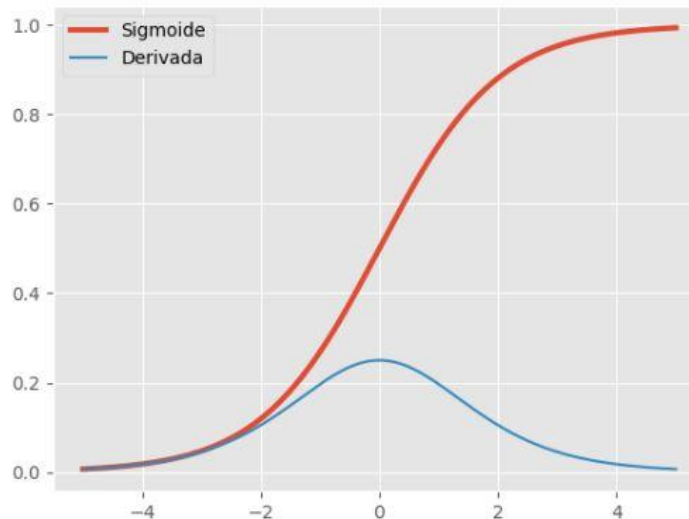


Figura 3. Função sigmoide logística e sua derivada (FACURE, 2017)

2.3.2. Redes Neurais Convolucionais

A rede neural convolucional (do inglês, *convolutional neural network* - CNN) (LECUN, 1989), também chamada ConvNet, é um tipo de arquitetura de RNP cujo princípio de operação foi inspirado pelo funcionamento do córtex visual de mamíferos, sendo particularmente projetada para o processamento de dados que possuem a estrutura de grades. As ConvNets, organizam os neurônios de suas camadas no formato de grades de n dimensões, com a quantidade de dimensões da grade podendo variar de acordo com a estrutura do dado de entrada (LECUN *et al.*, 2015; GOODFELLOW *et al.*, 2016; ALESHIN-GUENDEL, 2017).

Por exemplo, ao trabalhar com imagens, que podem ser consideradas como uma grade 2-D de pixels (matriz de pixels), a camada de entrada de uma CNN possuirá 3 dimensões (altura da imagem, largura da imagem e número de canais da imagem). Caso a imagem de entrada seja em preto e branco, o número de canais será igual a 1 e a imagem será representada apenas como uma única matriz de pixels (uma grade 2D). Em contrapartida, caso a imagem seja colorida, o número de canais será igual a 3, correspondente ao número de cores no sistema RGB (*Red* – Vermelho, *Green* – Verde e *Blue* – Azul). Neste caso, a camada de entrada representará a imagem colorida através de 3 matrizes de pixels empilhadas (uma grade 3D), onde cada matriz representa os valores dos pixels de um canal de cor da imagem. A Figura 4, ilustra como uma imagem colorida é representada pela camada de entrada de uma CNN.

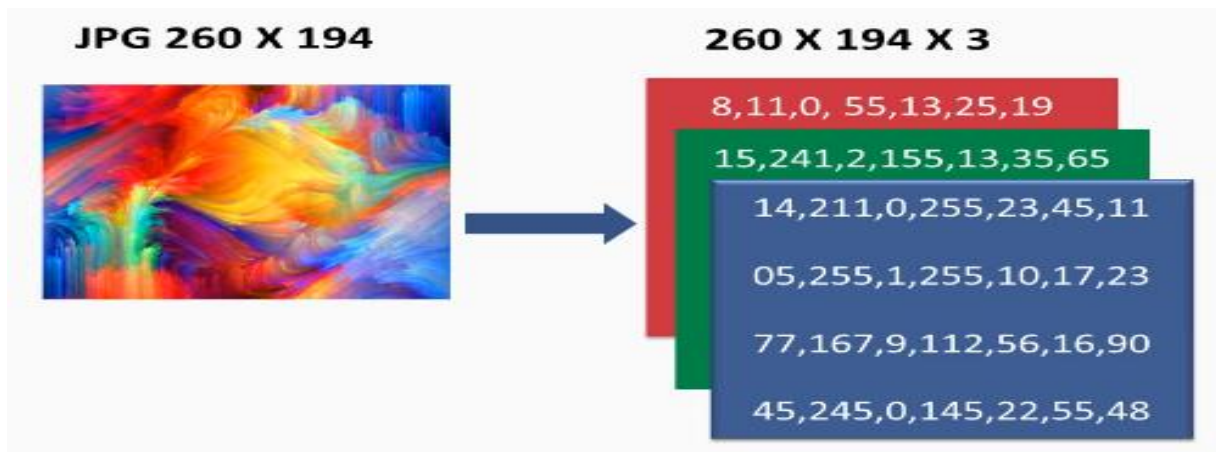


Figura 4. Exemplo uma imagem colorida sendo representada pela camada de entrada de uma CNN como grade 3D (3 matrizes de pixels empilhadas) (SAYEED *et al.*, 2020)

Comparando a MLP e a CNN, somente em questão de organização interna dos neurônios das camadas, é possível afirmar que a CNN é uma arquitetura mais flexível, que permite trabalhar de forma mais fácil e eficiente com diferentes tipos de estruturas de dados. Uma vez que, a MLP organiza os neurônios de suas camadas em formato de vetores, ou seja, mesmo que o dado de entrada possua uma estrutura com mais de uma dimensão, a MLP transformará esse dado em um vetor de uma dimensão, fazendo com que toda a informação espacial do dado de entrada seja perdida. Por outro lado, a CNN, ao trabalhar com grades de n dimensões, consegue preservar essa informação espacial que o dado de entrada multidimensional por natureza contém.

Visando ilustrar a diferença entre a MLP e a CNN, ao trabalharem com uma estrutura de dados 2D, como uma imagem, a Figura 5, do lado esquerdo, mostra com uma MLP simples, com 3 camadas, se estrutura para trabalhar com esse tipo de dado. Já o lado direito da Figura 5, mostra como uma ConvNet, organiza as suas camadas para lidar com o mesmo tipo de dado. Nesta ilustração, a camada de entrada vermelha representa a imagem, portanto, na ConvNet, sua largura e altura seriam as dimensões da imagem e a profundidade seriam os 3 canais RGB. Além disso, até a camada oculta da ConvNet, que será discutida na seção 2.3.2.3, também é formada por uma grade 3D.

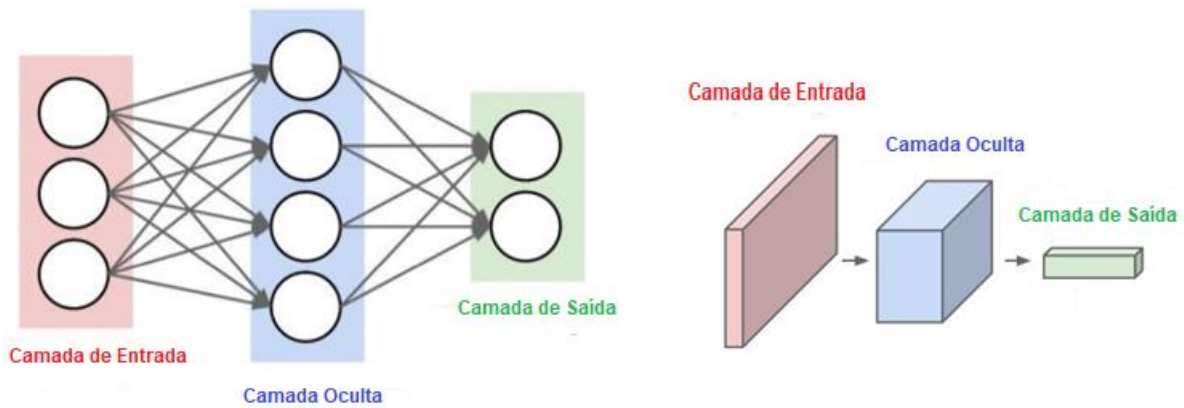


Figura 5. Esquerda: estrutura interna das camadas de uma MLP. Direita: estrutura interna das camadas de uma CNN (STANFORD, s.d).

Ademais, atualmente, a CNN é uma das arquiteturas de RNPs mais utilizadas, trabalhando com diferentes tipos de estruturas de dados, e se destacando na solução de diferentes tarefas que envolvem tais estruturas, podendo citar-se: processamento de sinais, séries temporais e linguagem natural (estruturas de dados 1D); processamento de imagens (estrutura de dados 2D); análise de vídeos (estrutura de dados 3D).

2.3.2.1. A Inspiração Biológica das Redes Convolucionais

Para compreender as ideias que fundamentam o funcionamento da CNN, é interessante entender um pouco do funcionamento do córtex visual. Os principais conceitos da CNN foram diretamente inspirados por algumas das primeiras descobertas dos pesquisadores David H. Hubel e Torsten Wiesel, sobre o modo de operação das estruturas do córtex visual.

Hubel e Wiesel em suas pesquisas mostram, que muitos neurônios do córtex visual possuem um pequeno campo receptivo local, o que significa que estes neurônios reagem apenas a estímulos visuais localizados em uma região limitada do campo visual. Os campos receptivos de diferentes neurônios podem se sobrepor e, juntos, formam todo o campo visual (HUBEL & WIESEL, 1959, 1968). Os pesquisadores também mostraram que alguns neurônios respondiam (eram ativados) apenas quando imagens de linhas horizontais apareciam em seu campo receptivo, enquanto outros neurônios reagiam apenas a linhas com orientações diferentes, de modo que foi constatado que era possível que dois neurônios poderiam ter o mesmo campo receptivo, mas reagiam a orientações de linhas diferentes. Além disso, também foi descoberto que alguns neurônios possuíam campos receptivos maiores e estes reagiam a padrões mais

complexos, que eram formados por combinações de padrões mais simples, como as linhas horizontais citadas (GÉRON, 2019).

A partir dessas observações, foi levantada a hipótese de que a informação no córtex visual era processada de forma hierárquica. Considerando que o córtex visual é dividido em várias estruturas, sendo o córtex visual primário (V1) formado por neurônios simples, com campos receptivos pequenos, que se ativam só quando detectam contornos ou linhas. Esses neurônios simples de V1, se conectam com neurônios mais complexos, com campos receptivos maiores, localizados no córtex visual secundário (V2), e estes, a partir das informações recebidas, são capazes de formar características mais complexas, como formas simples. Esse processo então prossegue, com as estruturas superiores do córtex visual formando características mais complexas a partir das características mais simples detectadas nas estruturas inferiores (GÉRON, 2019). A Figura 6 mostra esse esquema de processamento hierárquico do córtex visual, onde a combinação de características simples (arestas ou linhas) leva a características complexas, como faces.

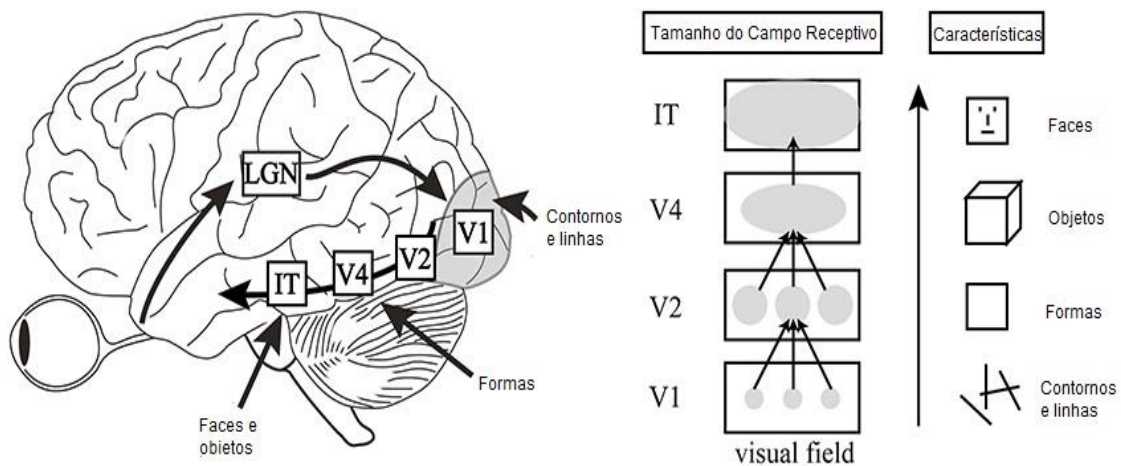


Figura 6. Descrição do funcionamento do córtex visual (HERZOG & CLARKE, 2014)

2.3.2.2. A Operação de Convolução

Fundamentalmente, na área do processamento de sinais digitais, a convolução é definida como uma operação matemática entre duas funções $f(x)$ e $g(x)$ para produzir uma terceira função, $h(x)$. Matematicamente, a convolução de duas funções é expressa como mostra a Eq (1). Já graficamente, a convolução dessas funções podem expressa como mostra a Figura 7. É importante salientar que na Eq (1) e na Figura 7, o símbolo * é a representação gráfica da operação de convolução entre as funções $f(x)$ e $g(x)$, e a integral é denominada de integral da convolução.

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(u)g(x - u)du \quad (1)$$

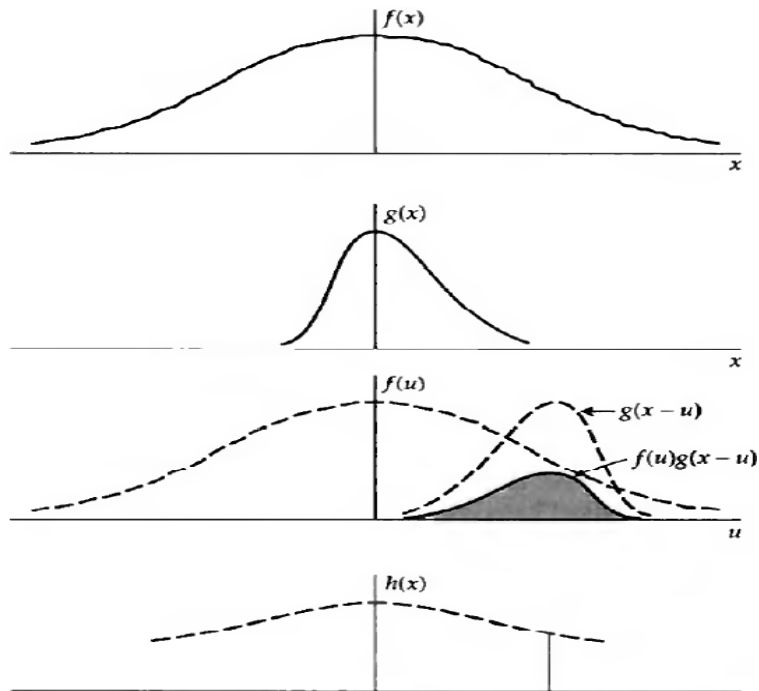


Figura 7. Representação gráfica da operação $h(x) = f(x) * g(x)$ (BRACEWELL, 1999)

A Eq (1) expressa a operação de convolução no tempo contínuo, contudo, nas ConvNets é utilizada a convolução no tempo discreto, sendo que as variáveis envolvidas agora são variáveis discretas, e a integral transforma-se em um somatório. Nesse contexto, a convolução discreta para dois vetores unidimensionais x e w é definida como $y = x * w$, em que o vetor x é a entrada (às vezes chamada de sinal) e w é chamado de filtro ou *kernel*. A Eq (2) mostra a definição matemática da convolução discreta em uma dimensão (RASCHKA & MIRJALILI, 2017).

$$x[n] * w[n] = \sum_{k=-\infty}^{\infty} s[k] \cdot w[i - k] \quad (2)$$

Para aplicar a Eq (2) nas CNNs, algumas modificações precisaram ser feitas na fórmula. Considerando que, em RNAs, trabalha-se com vetores de tamanho finito, para calcular corretamente a soma mostrada na Eq (2), supõe-se que x e w sejam preenchidos com zeros. Contudo, isso resultaria em um vetor de saída y que também possuiria tamanho infinito. Sendo assim, como isso não é útil em situações práticas, x é

preenchido apenas com um número finito de zeros. Esse processo é chamado de preenchimento de zeros (*zero-padding*), ou simplesmente preenchimento (*padding*).

A partir desta premissa, considerando que o vetor de entrada x e o filtro w tenham, respectivamente, n e m elementos, onde $m \leq n$, a Eq (2) será agora alterada para a Eq (3), e nesta x e w são indexados em direções inversas. Em virtude disso, é possível inverter um desses vetores, x ou w , de modo que, é possível calcular o seu produto escalar. Sendo assim, invertendo o filtro w , o filtro rotacionado w^r é obtido. Então, o produto escalar $x[i:i+m].w^r$ é calculado para obter um elemento $y[i]$, onde $x[i:i+m]$ é uma parte de x com o tamanho m . Esta operação é repetida como em uma abordagem de janela deslizante (*sliding window*) para obter todos os elementos de saída. Um exemplo deste processo é exibido na Figura 8, neste caso, o vetor x não possui *padding*, e o filtro rotacionado w^r , a cada iteração, é deslocado duas células para a direita. Este deslocamento é chamado de *stride*, sendo um dos hiperparâmetros de uma convolução, no exemplo da Figura 8, *stride* = 2. Ademais, observe-se que o valor do *stride* deve ser um número positivo menor que o tamanho do vetor de entrada (RASCHKA & MIRJALILI, 2017).

$$x[n] * w[n] = \sum_{k=0}^{k=m-1} x^p[i+m-k] w[k] \quad (3)$$

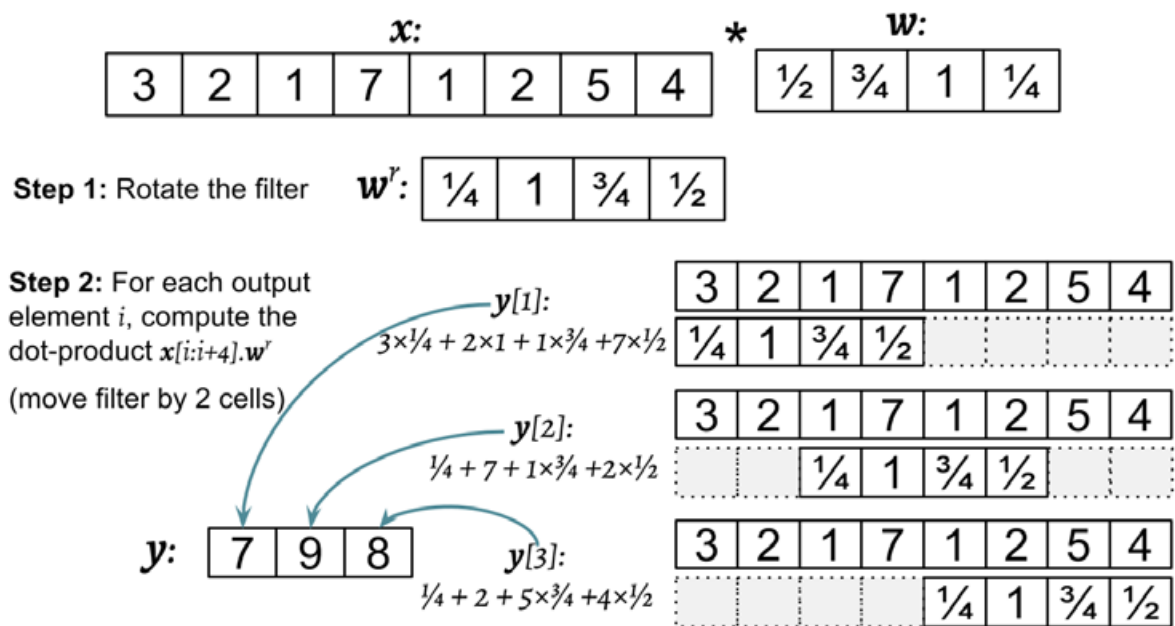


Figura 8. Etapas no processo de convolução de dois vetores (RASCHKA & MIRJALILI, 2017)

A operação de convolução também pode ser aplicada em estruturas de dados 2D, como matrizes. Considerando uma matriz de entrada X_{m_1, m_2} e matriz de filtro W_{n_1, n_2} , onde $m_1 \leq n_1$ e $m_2 \leq n_2$, então a matriz $Y = X * W$ é o resultado da convolução 2D de X com W . Sendo este conceito matematicamente definido pela Eq (4), que é similar à formulada da convolução 1D, de modo que todos os conceitos citados: preenchimento de zero, rotação de filtro e a movimentação em *strides*, também são aplicáveis à convolução 2D.

$$Y = X * W = Y[i, j] = \sum_{k_1=-\infty}^{+\infty} \sum_{k_2=-\infty}^{+\infty} X[i - k_1, j - k_2] W[k_1, k_2] \quad (4)$$

Um exemplo de convolução 2D, pode ser apresentado considerando:

- Matriz de Entrada $X = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 2 & 0 \\ 0 & 5 & 0 & 1 & 0 \\ 0 & 1 & 7 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
- Matriz de Filtro $W = \begin{bmatrix} 0,5 & 0,7 & 0,4 \\ 0,3 & 0,4 & 0,1 \\ 0,5 & 1 & 0,5 \end{bmatrix}$

Neste caso, a matriz X possui *padding* = (1,1), significando que uma camada de zeros complementa cada lado da matriz de entrada. Deste modo, a Figura 9 ilustra a operação de convolução $X * W$.

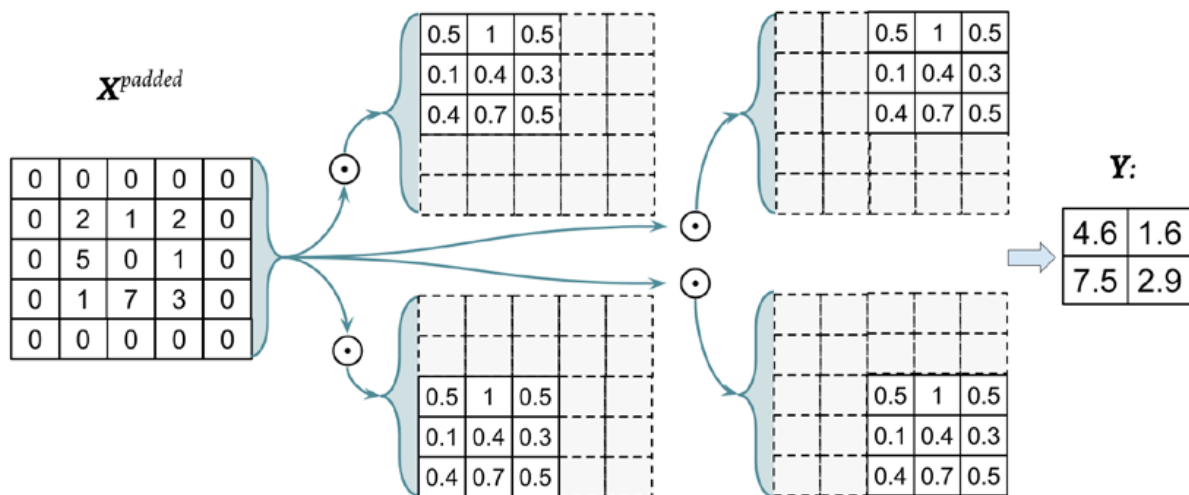


Figura 9. Operação de convolução 2D entre a entrada e a matriz rotacionada (RASCHKA & MIRJALILI, 2017)

2.3.2.3. A Camada Convolutiva

A estrutura principal das CNNs é a camada oculta convolutiva, geralmente a primeira camada oculta deste tipo de rede. Sendo assim, uma camada convolutiva é formada por diversas grades de neurônios empilhadas, sendo estas grades chamadas mapas de características (do inglês, *feature maps*). Cada mapa de características é gerado a partir da operação de convolução entre a camada anterior e um conjunto de pesos, chamado de filtro ou *kernel*. No exemplo da Figura 10, é mostrado o processo para gerar um mapa de características para uma entrada com estrutura 2D. Neste caso, um filtro 2D também é utilizado e a operação de convolução discreta 2D é aplicada.

O objetivo da convolução entre o filtro e entrada é gerar um mapa de características que destaque características relevantes da entrada. Esse conceito é melhor entendido de forma visual ao utilizar imagens com exemplo. Ao realizar a convolução de uma imagem com uma matriz de valores, uma imagem resultante é gerada e, esta imagem gerada, representa a imagem original com uma característica destacada, como mostra a Figura 11. Por causa desse fato, que o conjunto de pesos também é chamado de filtro na CNN, pois, por intermédio da convolução, o conjunto de pesos filtra toda a imagem em busca de uma característica específica (VASILEV *et al.*, 2019). Existe todo um campo de pesquisa que estuda a criação de filtros para buscar características específicas em uma imagem.

Ademais, a extração de características que ocorre na camada convolutiva pode ser analisada a partir da inspiração no neurônio biológico. Embora os neurônios em um

mapa de características de uma CNN se conectem com a camada anterior compartilhando o mesmo conjunto de pesos, cada neurônio, na verdade, se conecta a uma região diferente da camada anterior. Ou seja, cada neurônio tem o seu próprio campo receptivo local, seguindo o mesmo princípio de conexão utilizado pelos neurônios biológicos presentes no córtex visual.

No contexto do aprendizado na CNN, como citado, os filtros são os conjuntos de pesos, sendo assim, ao se determinar a quantidade de filtros, a própria rede através do seu processo de aprendizado é responsável por encontrar os valores que formam esses filtros, de modo que estes filtros sejam capazes de detectar as características mais relevantes para a resolução de uma determinada tarefa, com base em um determinado conjunto de dados de treinamento. Entretanto, os valores dos filtros são os únicos parâmetros aprendidos automaticamente durante o processo de treinamento na camada convolucional, com isso, o tamanho dos filtros, o número de filtros, a *padding* e o *stride* são hiperparâmetros que precisam ser definidos antes do início do processo de treinamento. Portanto, é importante ressaltar que ao se definir a quantidade de filtros de uma camada convolucional, concomitantemente, também está sendo definida a quantidade de mapas de características que vai formar esta camada (YAMASHITA *et al.*, 2018).

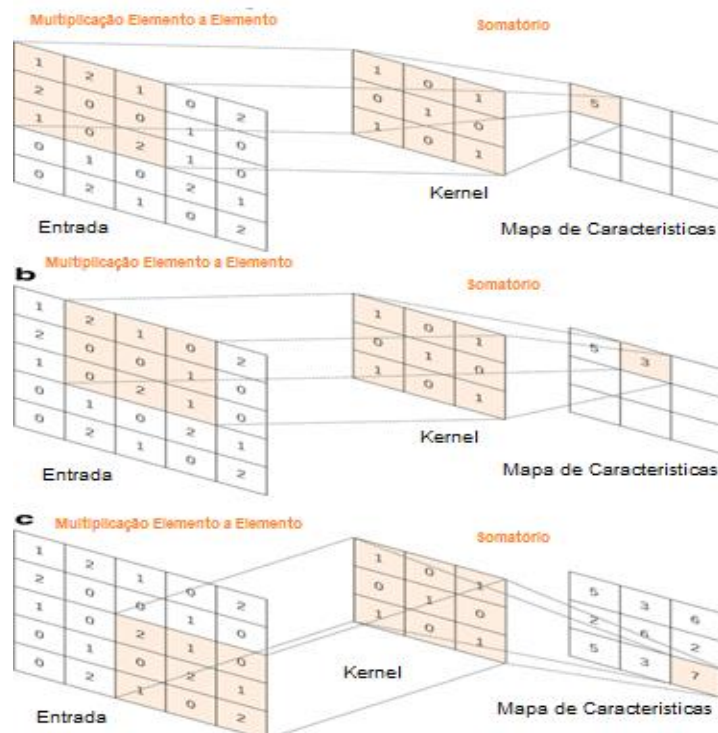


Figura 10. Processo de geração de uma mapa de características, através da convolução do filtro com a entrada (YAMASHITA *et al.*, 2018)







Operation	Kernel	Image result
Edge detection	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 5 x 5 (approximation)	$\frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	
Unsharp masking 5 x 5 Based on Gaussian blur with amount as 1 and threshold as 0 (with no image mask)	$\frac{-1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & -476 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

Figura 11. Exemplos dos resultados ao convoluir diferentes filtros com uma imagem (ALBAWI, 2018)

2.3.2.4. Camada de Subamostragem

A outra camada oculta que geralmente é utilizada em ConvNets é a camada de subamostragem (do inglês, *pooling layer*), a qual é usualmente colocada após uma camada convolucional, com os seus neurônios também usando campos receptivos locais para se conectar. As funções da camada de subamostragem são: reduzir o tamanho espacial dos mapas de características da camada convolucional, agilizando o processo de treinamento; deslocar e distorcer os mapas de características da camada convolucional, para não permitir que a CNN fique dependente da posição exata de uma característica relevante para assim detectá-la (YAMASHITA *et al.*, 2018). Para cumprir essas funções, a camada de subamostragem utiliza operações de subamostragem, com as principais opções desse tipo de operação sendo:

- **Max-Pooling:** É uma operação matemática que funciona retirando o maior valor de uma região, de tamanho pré-determinado, de uma entrada. Por exemplo, na situação da Figura 12, um filtro de *max pooling* de tamanho 2×2 , com um *stride* 2 é usado. Com isso, à medida que o filtro percorre o mapa de características, este vai retirando o maior valor de

cada região, e ao final, um mapa de características de tamanho reduzido é gerado.

- **Average-Pooling:** É uma operação matemática que funciona retirando a média dos valores de uma região, de tamanho pré-determinado, de uma entrada. Por exemplo, na situação da Figura 12, um filtro de *average-pooling* de tamanho 2×2 , com um *stride* 2 é usado. Com isso, à medida que o filtro percorre o mapa de características, este vai retirando a média dos valores de cada região, e ao final, um mapa de características de tamanho reduzido é gerado.

Geralmente, a operação de *max pooling* é mais utilizada em camadas de subamostragem em arquiteturas de CNN (VASILEV *et al.*, 2019), contudo, não há uma regra de ouro que defina qual é a melhor operação de subamostragem. Em virtude disso, a seleção do tipo de operação é empírica, dependendo do tipo de arquitetura de CNN e tarefa na qual esta vai ser utilizada.

Um ponto que é importante notar, é que a camada de subamostragem realiza a operação de convolução selecionada em todos os mapas de características da camada convolucional anterior, ou seja, a camada de subamostragem gera como saída o mesmo número de mapas de características da camada anterior, contudo, o tamanho desses mapas é reduzido. Além disso, a camada de subamostragem não possui parâmetros para serem aprendidos, uma vez que esta camada é usada somente para fazer essa redução no tamanho dos mapas de características, sendo assim, os únicos hiperparâmetros que precisam ser definidos para essa camada são: tamanho dos filtros de subamostragem e o *stride*.

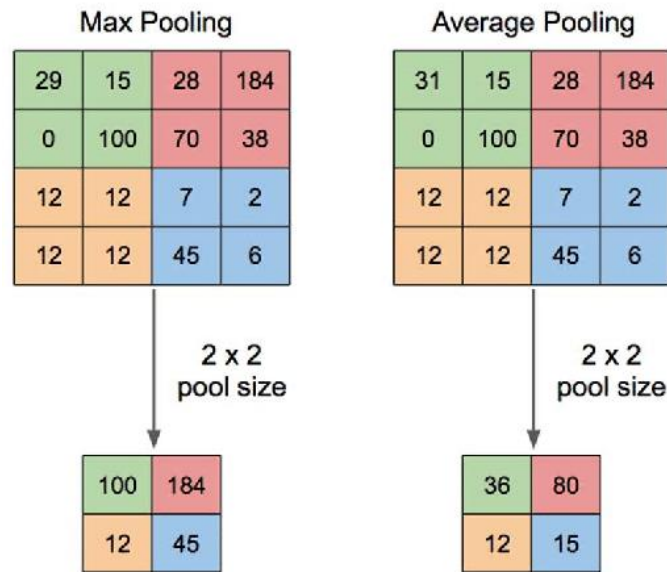


Figura 12. Esquerda: Operação de max pooling. Direita: Operação de average pooling (MUHAMAD *et al.*, 2019)

2.3.3. Redes Neurais Recorrentes

A rede neural recorrente (do inglês *recurrent neural network* - RNN) é uma arquitetura de RNP elaborada especialmente para processar dados sequências, chamados tradicionalmente de sequências. Assim como a ConvNet é uma arquitetura especializada em processar uma grade de valores X, como uma imagem. A RNN é uma arquitetura especializada em processar uma sequência de valores ($x^1 \dots x^t$), onde os elementos da sequência aparecem em uma determinada ordem e não são independentes entre si. Além disso, assim como as ConvNets podem ser facilmente dimensionadas para imagens com de tamanho variável. As RNNs podem ser dimensionadas para sequências longas, sendo que a maioria das RNNs também podem processar sequências de tamanho variável (GOODFELLOW *et al.*, 2016).

2.3.3.1. Diferentes Categorias de Modelagem de Problemas Sequenciais

Dada a arquitetura da RNN, esta pode ser utilizada para modelar e processar diferentes categorias de dados sequenciais. Sendo estas categorias definidas a partir de como as entradas e saídas do conjunto de dados sequencial são formatadas. Portanto, ao observar a Figura 13 é possível definir as categorias principais de dados sequenciais:

- **Muitos-para-Um (*Many-to-One*):** As entradas são sequências, enquanto as saídas não são sequências, mas sim vetores de tamanho fixo. Esse tipo de categoria de dado sequencial é utilizado para problemas de

classificação de sequências e previsão de sequências. Alguns dos problemas que estão nessa categoria são:

- Detecção de anomalia: É um problema de classificação de sequências já que consiste em prever se uma determinada sequência de observações é uma anomalia ou não.
- Previsão de Ações de Bolsa de Valores: Esse problema pode ser categorizado como um problema de previsão de sequências, pois a sua premissa envolve a utilização de uma sequência de valores passados, como por exemplo os preços de ações, para prever o próximo valor dessa sequência.
- **Um-para-Muitos (*One-to-Many*):** As entradas são vetores de tamanho fixo, enquanto as saídas não são sequências. Um exemplo desse tipo de problema é a geração de legendas em imagem, onde a entrada é uma única imagem, e a saída é uma sequência de palavras que descrevem a imagem.
- **Muitos-para-Muitos (*Many-to-Many*):** As entradas e saída são sequências, por isso muitas vezes os problemas que se enquadram nessa categoria são chamados de problemas de sequências-para-sequências (do inglês, *sequence-to-sequence*). Os problemas de sequências-para-sequências podem ser estruturados de duas formas principais, em virtude disso a categoria muitos-para-muitos pode ser subdividida da seguinte maneira:
 - **Muitos-para-Muitos Sincronizado (*Many-to-Many Synchronized*):** As sequências de entrada e saída são sincronizadas, ou seja, as sequências de entrada e de saída possuem o mesmo tamanho. Um exemplo é a classificação de vídeos, onde a sequência de entrada é formada por *frames* de um vídeo, e a sequência de saída são os rótulos atribuídos a cada um desses *frames*.
 - **Muitos-para-Muitos Dessincronizado (*Many-to-Many Desynchronized*):** As entradas e saídas sequenciais não são sincronizadas, sendo que a sequência de entrada e de saída não

possuem o mesmo tamanho. O principal exemplo é a tradução, onde a entrada é uma sequência tamanho x de palavras em um idioma, enquanto a saída é uma sequência de palavras de tamanho y em outro idioma.

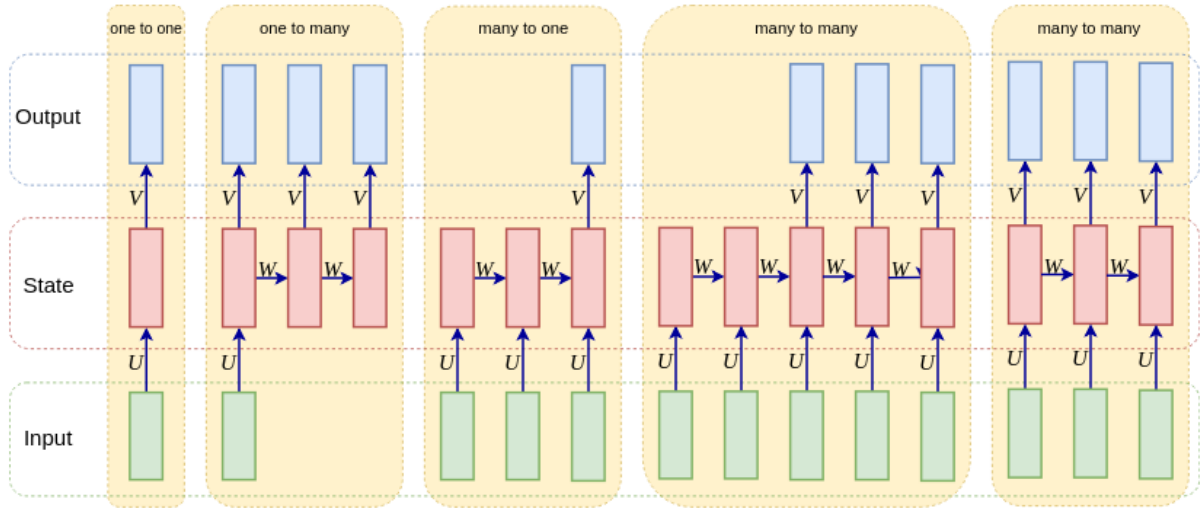


Figura 13. As categorias de dados sequencias (VASILEV *et al.*, 2019).

2.3.3.2. Estrutura e o Fluxo de Informações em Redes Recorrentes

As redes neurais recorrentes recebem esse nome devido ao fato de aplicarem a mesma função em uma sequência recorrente, de modo que uma RNN pode ser definida como de relação de recorrência, mostrada na Eq (5):

$$s_t = f(s_{t-1}, x_t) \quad (5)$$

Na Eq (5), f é uma função diferenciável, s_t é um vetor de valores, chamado estado interno da rede (no passo t) e x_t é a entrada da rede no passo t , em contraste, com arquiteturas de rede progressiva, onde o estado depende apenas da entrada atual (e dos pesos da rede). Na arquitetura recorrente, o estado s_t é uma função da entrada atual e do estado anterior s_{t-1} , sendo que, s_{t-1} pode ser entendido como o resumo de todas as entradas anteriores. Portanto, a relação de recorrência na RNN define como o estado evolui passo a passo sobre uma sequência por meio de um loop de feedback sobre os estados anteriores, de forma que, o fluxo de informações em passos de tempo adjacentes na camada oculta permite que a RNN tenha uma memória de eventos passados (VASILEV *et al.*, 2019).

O fluxo de informações na RNN, geralmente é exibido como um *loop de feedback*, também conhecido como uma borda recorrente (do inglês, *recurrent edge*) na notação gráfica. Para melhor examinar a arquitetura da RNN e seu fluxo de informações, graficamente, a borda recorrente pode ser desdobrada (*unfold*) (RASCHKA & MIRJALILI, 2017). Sendo assim, a Figura 14 ilustra graficamente o fluxo de informações em uma RNN, com a borda recorrente desdobrada.

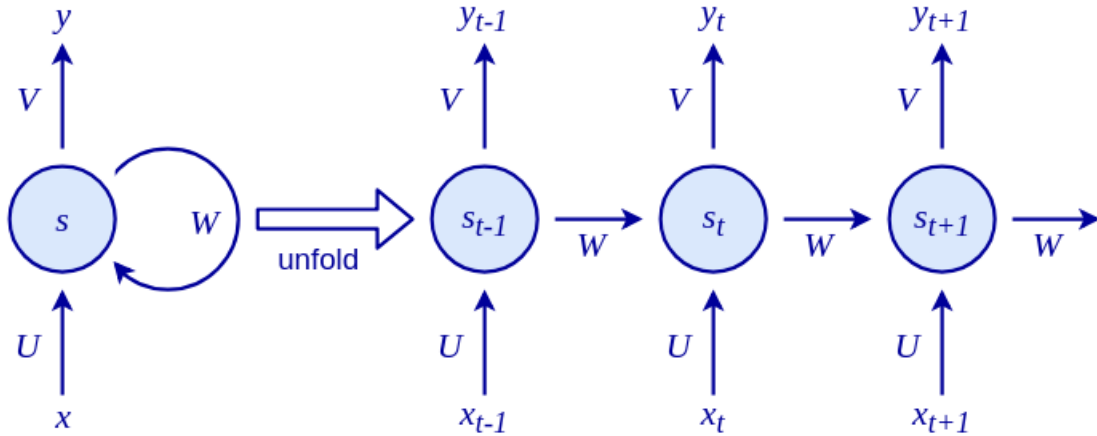


Figura 14. Fluxo de Informação em uma RNN (VASILEV *et al.*, 2019)

Como a Figura 14 mostra, a arquitetura da RNN tem três conjuntos de pesos: U que transforma a entrada x_t no estado atual s_t ; W que transforma o estado anterior s_{t-1} no estado atual s_t ; e V que mapeia o novo estado atual s_t para a saída y_t . Os conjuntos U , V e W aplicam transformação linear (soma ponderada) sobre suas respectivas entradas. A equação do estado s_t , considerando os conjuntos de pesos, pode ser definida pela Eq (6), enquanto a equação da saída y_t pode ser definida pela Eq (7) (VASILEV *et al.*, 2019).

$$s_t = f(s_{t-1}W + x_tU) \quad (6)$$

$$y_t = s_tV \quad (7)$$

Em relação ao fluxo de informação, ao receber uma sequência como entrada, no primeiro passo ($t = 0$), as unidades ocultas são inicializadas com zeros ou pequenos valores aleatórios. Já em um passo temporal, onde $t > 0$, as unidades ocultas recebem dois valores distintos de entrada, um vindo de x_t e outro de s_{t-1} (RASCHKA & MIRJALILI, 2017). O modo como a RNN é arquitetada, usando esta relação de

recorrência, faz com que cada estado seja dependente de todos os cálculos anteriores. Isso acaba atribuindo à RNN uma “memória”, porque as informações dos estados s foram construídas baseadas nas informações em passos anteriores, de modo que, em teoria, a RNN poderia memorizar longas sequências, contudo, essa não é a realidade para todos os modelos de RNN, devido ao problema de desaparecimento de gradiente.

2.3.4. Redes Neurais Autocodificadoras

A RNA autocodificadora, originalmente chamada de rede associativa, é um tipo específico de RNA, não sendo exatamente uma arquitetura por si só, mas sim uma forma de projetar e treinar uma RNA progressiva, convolucional ou recorrente, para que esta aprenda a reproduzir os dados de entrada como dados de saída. O modelo clássico de *Autoencoder* (AE) (KRAMER, 1992), mostrado na Figura 15, é dividido em duas estruturas, a primeira é o codificador, composto pela camada de entrada e uma camada oculta (às vezes chamada de camada de gargalo ou camada codificada), que codifica a entrada original, comprimindo-a para um espaço com dimensionalidade inferior, com o objetivo de aprender somente as características mais relevantes necessárias para reconstruí-lo. A segunda estrutura é o decodificador, composto pela camada de gargalo e pela camada de saída, que tenta decodificar a entrada, reconstruindo-a do seu espaço dimensional inferior de volta à representação original. Durante o processo de treinamento do AE, o erro entre os valores originais de entrada e os valores reconstruídos de saída é calculado e os pesos da rede são ajustados para minimizar esse erro; em virtude disso, este é chamado de erro de reconstrução (KRAMER, 1992; HINTON & SALAKHUTDINOV, 2006).

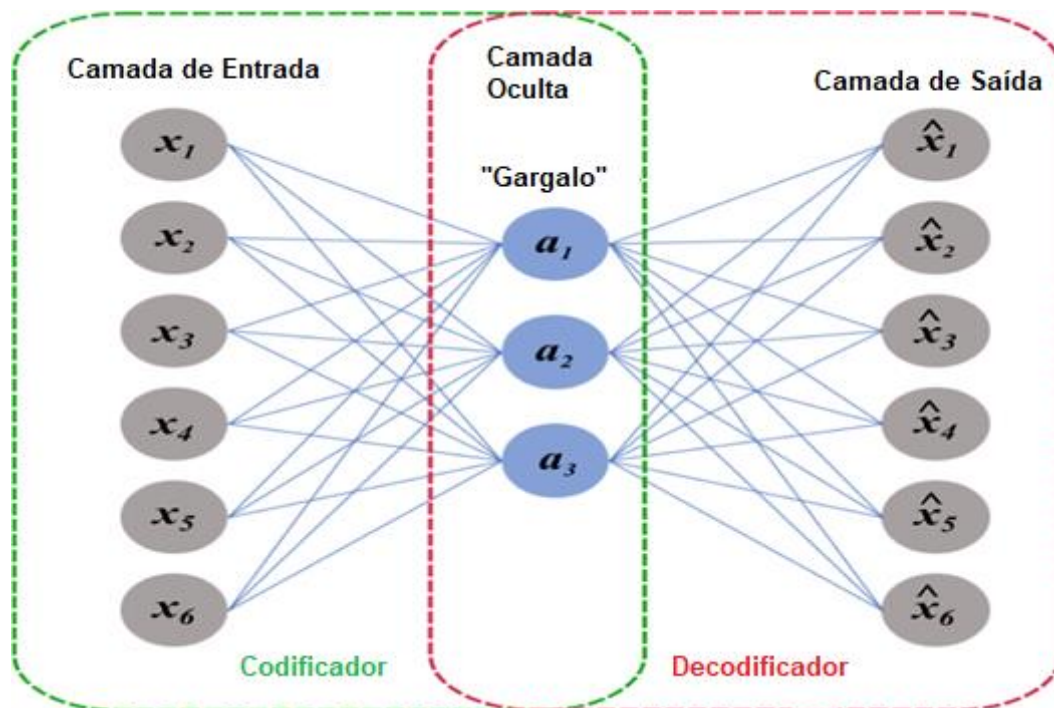


Figura 15. Arquitetura de autoencoder clássico (SAGHEER & KOTB, 2019)

Atualmente, o *deep autoencoder* (DAE) é o modelo mais popular de AE e ao contrário do AE clássico que possui apenas uma estrutura codificadora e decodificadora, os DAEs são formados por vários estágios de codificação e decodificação, sendo esses estágios representados por uma sequência de camadas de codificação, seguida por uma pilha de camadas de decodificação (ZHOU *et al.*, 2014). Ao empregar múltiplas camadas codificadoras e decodificadoras, um DAE tem a capacidade de discriminar e representar características complexas de conjuntos de dados, de maneira mais eficaz (ZHOU & PAFFENROTH, 2017).

Dada capacidade dos AEs aprenderem a reconstruir os dados treinados originais com um erro mínimo de reconstrução, quando um modelo de AE, após o treinamento, é confrontado com uma entrada que se desvia de alguma maneira dos dados utilizados durante o treinamento, um alto erro de reconstrução será gerado pelo modelo. Em virtude disso, os AEs são amplamente empregados na tarefa de detecção de novidade. Pois, definindo um valor limiar para o erro de reconstrução, o AE pode ser utilizado para indicar se a entrada recebida é desconhecida, ou seja, possui um padrão diferente daquele observado no conjunto de treinamento.

Sendo assim, no contexto da detecção de novidade, existem diferentes abordagens na utilização do AE, por exemplo, o AE pode ser treinado com um conjunto de classes para assim aprender a reconstruí-las. Na área nuclear, essa abordagem foi

utilizada por Yang *et al.*, (2020), treinando um modelo de AE, com arquitetura recorrente, com um conjunto de acidentes nucleares, de modo a utilizar o modelo para indicar quando a entrada, recebida por um sistema de classificação de acidentes nucleares, não fazia parte dos acidentes conhecidos pelo sistema.

Outra abordagem é usar o AE como uma OC-NN, ou seja, treinar o AE para aprender a reconstruir somente uma única classe. Esta abordagem, também já foi analisada na área nuclear (PINHEIRO *et al.*, 2020), nesta pesquisa múltiplos *one class* DAEs com arquitetura progressiva foram utilizados para atribuir a capacidade de detecção de novidade a um sistema de identificação de acidente em usinas nucleares. Neste caso, cada *one class* DAE foi treinado somente com dados de um dos acidentes contemplados pelo sistema. Embora, ambas as abordagens sejam validas, o uso de OC-NNs para este tipo de tarefa vem alcançando melhores resultados(GUTOSKI *et al.*, 2017; CHALAPATHY *et al.*, 2018; WEI *et al.*, 2018; OZA & PATEL, 2019; PERERA *et al.*, 2019).

CAPÍTULO 3

DESCRIÇÃO DO SISTEMA PROPOSTO

3.1. DESCRIÇÃO DOS PRINCIPAIS PROBLEMAS ENFRENTADOS PELO SISTEMA PROPOSTO

3.1.1. Classificação de Séries Temporais

O principal desafio do PIDAN, é resolver a tarefa de classificação de acidentes de uma usina nuclear. Sendo que, cada acidente é representado pelo comportamento temporal de diversas variáveis de estado da usina, como: vazão de vapor, temperatura na perna quente; temperatura na perna fria; nível do pressurizador; entre outras, durante a ocorrência do acidente. Visando ilustrar essa definição, a Figura 16 mostra o comportamento temporal, de 61 segundos, de 16 variáveis de estado (vazão no núcleo (%); temperatura na perna quente; temperatura na perna fria; vazão no núcleo (kg/s); nível no gerador de vapor – faixa larga; nível no gerador de vapor – faixa estreita, pressão no gerador de vapor; vazão de água de alimentação; vazão de vapor; vazão no circuito primário; pressão no sistema primário; potência térmica; potência nuclear; margem de sub-resfriamento; nível do pressurizador e temperatura média do primário) durante o acidente de BLACKOUT em um PWR, enquanto a Figura 17 mostra o comportamento temporal das mesmas variáveis durante o acidente LOCA no mesmo reator.

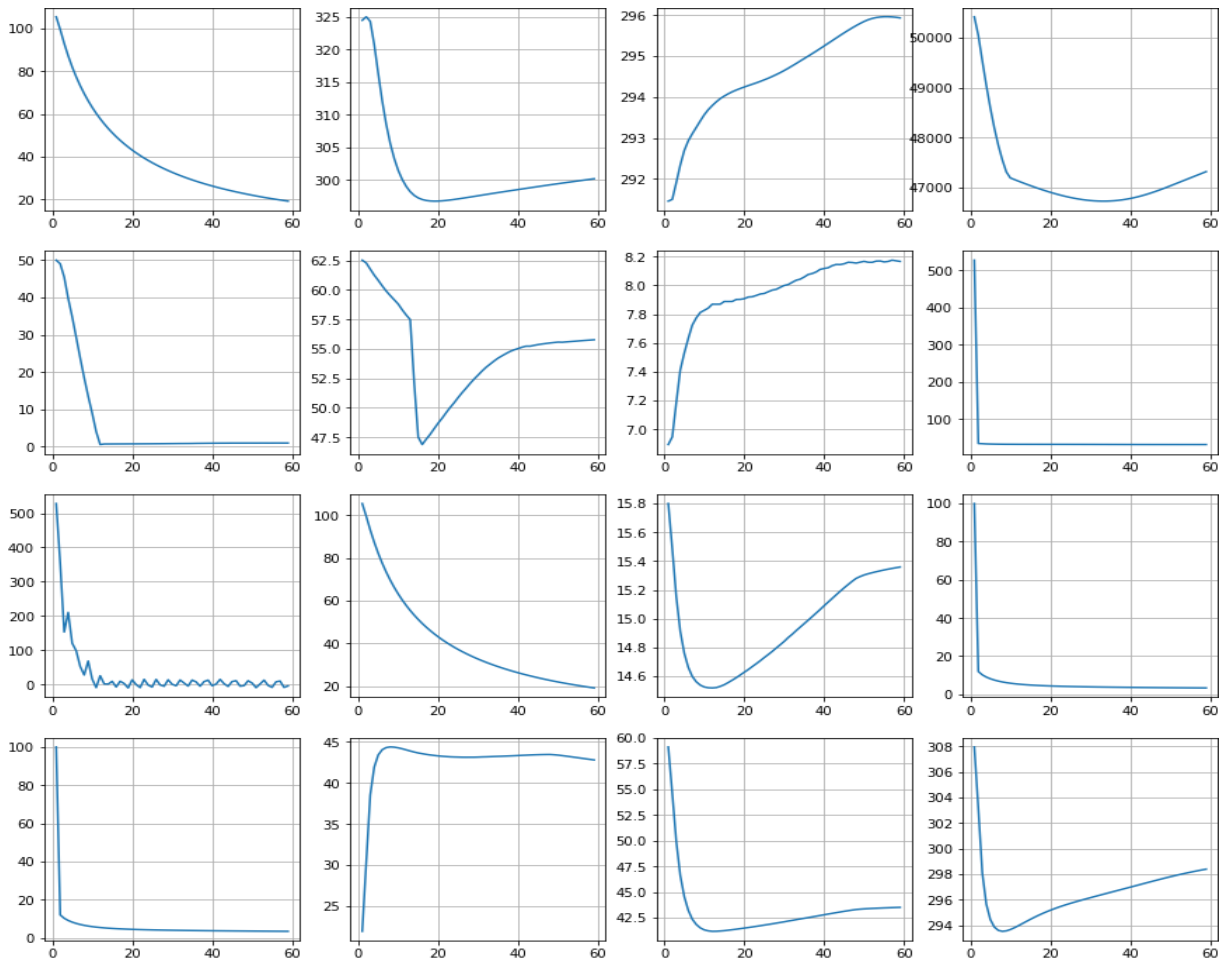


Figura 16. Evolução temporal de 16 variáveis de estado durante um acidente BLACKOUT em um PWR (De autoria própria).

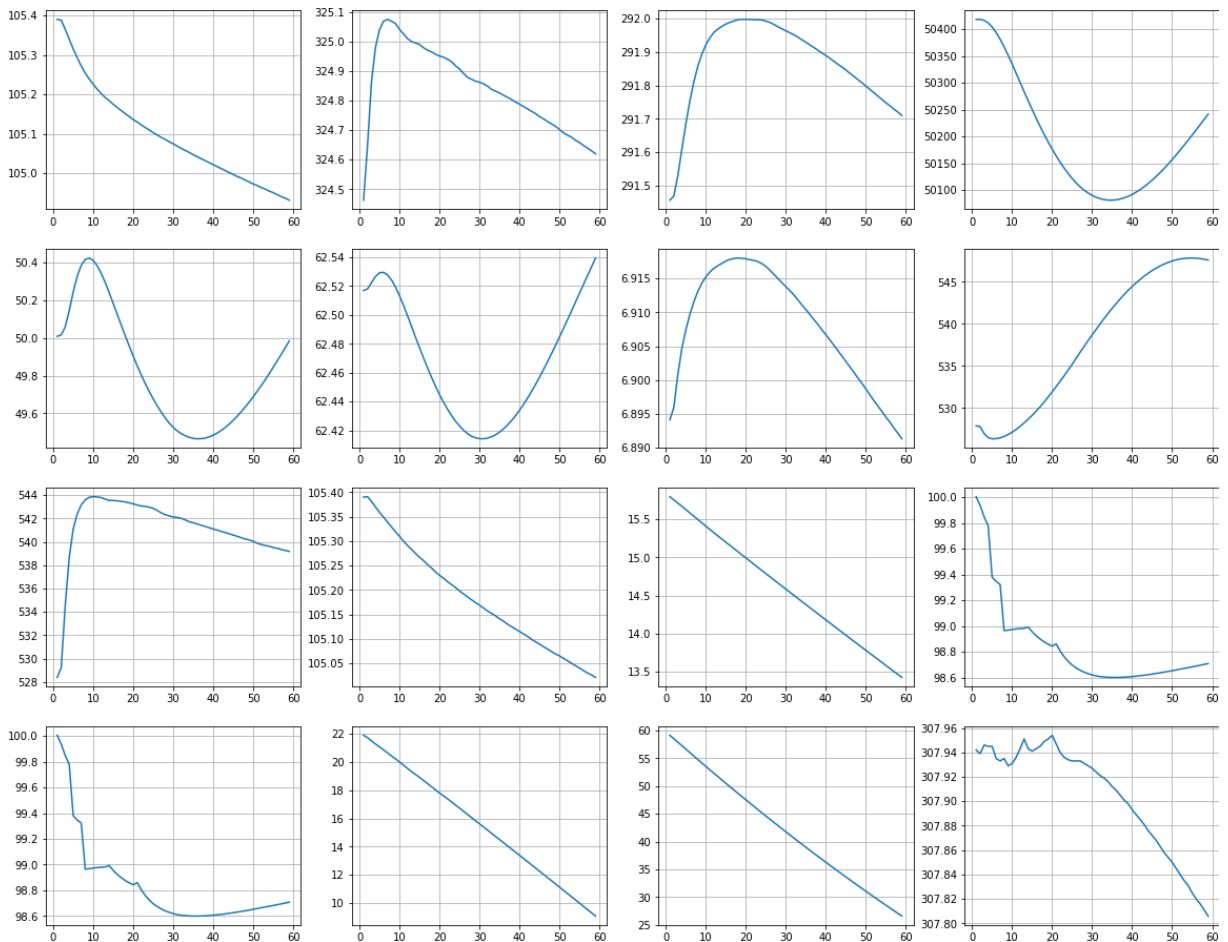


Figura 17. Evolução temporal de 16 variáveis de estado durante um acidente LOCA

(De autoria própria).

Comparando às duas figuras, é possível perceber que há uma clara diferença no comportamento da maioria das variáveis durante cada um dos acidentes (BLACKOUT e LOCA). Deste modo a resolução dessa tarefa se dá através de abordagens capazes de classificar corretamente os acidentes, a partir da evolução temporal das variáveis que os descrevem. Portanto, dado esta natureza do problema de classificação de acidentes nucleares, este se enquadra em uma classe mais abrangente de problema, que é a de CST.

A CST pode ser resumida de forma simples, como a tarefa onde se deseja classificar uma ou múltiplas séries temporais, em uma de duas ou várias classes predefinidas. Posto isto, um problema de CST pode lidar com séries temporais univariáveis ou multivariáveis, de forma que uma série temporal univariável, $X = [x_1, x_2 \dots x_T]$, é definida como uma sequência de valores reais ordenados em instantes de tempo, sendo o comprimento de X igual ao número de valores reais T . Já uma série temporal multivariável, $X = [X^1, X^2 \dots X^M]$, consiste em M diferentes séries temporais

univariadas, com $X^i \in \mathbb{R}^T$ (FAWAZ *et al.*, 2019; SUSTO *et al.*, 2018). Ademais, a tarefa de CST é considerada um problema extremamente desafiador e amplamente pesquisado, abrangendo uma vasta gama de aplicações em diversos campos de estudo. Nas últimas décadas, o interesse na CST aumentou ainda mais devido à maior disponibilidade de dados temporais. Em virtude disso, diversos novos métodos estão sendo utilizados para lidar com a tarefa de CST, entre as quais estão as RNPs.

Os estudos envolvendo técnicas de aprendizado profundo para resolver problemas de CST ainda são poucos, contudo, os resultados apresentados são promissores. Nesse contexto, um trabalho de grande relevância é o de Fawaz *et al.*, (2019), onde foi realizado, até o momento, o maior estudo empírico de RNPs para a tarefa de CST. O estudo envolveu modelos específicos das três arquiteturas principais de RNP: progressiva, convolucional e a recorrente. No total foram usados 9 modelos de RNPs, um baseado na arquitetura progressiva, um na arquitetura recorrente e 7 baseados na arquitetura convolucional. Alguns dos 9 modelos de RNP (ZHENG *et al.*, 2014, 2016; CUI *et al.*, 2016; TANISARO e HEIDEMANN, 2016; LE GUENNEC *et al.*, 2016; ZHAO *et al.*, 2017; WANG *et al.*, 2017; SERRÁ *et al.*, 2018) já haviam sido publicados em trabalhos científicos prévios, onde demonstraram um bom desempenho em problemas de CST.

Fawaz *et al.*, (2019) compararam o desempenho dos 9 modelos de RNP, utilizando 97 conjuntos de dados tradicionais para avaliação do desempenho geral para a CST (BAGNALL *et al.*, 2018; DAU *et al.*, 2019; YANPING *et al.*, 2015), cada um desses conjuntos representava um problema de CST diferente. Dentre os modelos, a rede totalmente convolucional (do inglês, *fully convolution network* - FCN) foi um dos modelos de RNP que demonstrou os melhores resultados, ou seja, obteve uma maior taxa de exatidão na classificação na maior parte dos conjuntos de dados. Além disso, posteriormente, os modelos de RNP foram comparados com métodos clássicos usados para a resolução de problemas de CST, com os mesmos 97 conjuntos de dados sendo usados para a comparação. O resultado foi que o desempenho das RNPs foi superior na maior parte dos conjuntos de dados, mostrando que ao utilizar modelos de aprendizagem profunda, como a FCN, é possível alcançar desempenho do nível do estado da arte em problemas de CST.

3.1.2. Detecção de Novidade

O problema de detecção de novidade (do inglês, *novelty detection*) pode ser definido como a identificação de “novidades”, que são padrões de dados novos, desconhecidos ou não observados, aos quais um sistema de aprendizado de máquina, como uma RNP, não foi exposto durante o seu treinamento (KWON *et al.*, 2019; MARKOU & SINGH, 2003). A capacidade de detectar novidade é imprescindível para um sistema de classificação ou identificação baseado em técnicas de aprendizado de máquina; sem essa funcionalidade o sistema não é considerado realmente completo (MARKOU & SINGH, 2003). Pois, dada a característica de mundo aberto da grande maioria dos problemas de classificação, independentemente da qualidade do conjunto de treinamento utilizado para treinar um modelo de aprendizado de máquina, este ainda sim é finito, sendo praticamente impossível assegurar, que o modelo treinado, mesmo tendo uma elevada capacidade de generalização e operando em escopo delimitado, consiga fornecer uma resposta correta para todas as situações apresentadas durante a sua operação no mundo real.

Portanto, considerando um sistema para classificação de acidentes em usinas nucleares, baseado em redes neurais, atribuir ao sistema a capacidade de detectar novidades é ainda mais necessário. Tendo em vista que uma rede neural, ao ser treinada para classificar um conjunto de classes, ao receber uma entrada, vai atribuir esta a uma das classes conhecidas, mesmo se o nível de confiabilidade dessa classificação for baixo. Sendo assim, ao desenvolver um sistema de suporte a decisão dos operadores da sala de controle, cujo classificador é um modelo de rede neural, é vital que uma abordagem para a detecção de novidade seja utilizada, de modo a validar a resposta do classificador, garantindo que a rede não classifique um padrão desconhecido como um pertencente a uma de suas classes conhecidas. Nesse contexto, muitas vezes na literatura, atribuir a um sistema, que usa rede neural para classificação, a capacidade de detectar novidades, significa fornecer a esse sistema a habilidade de gerar a resposta “Não Sei”.

Existem diversas abordagens para lidar com a tarefa de detecção de novidades, e assim como vem acontecendo com diversas outras tarefas, nos últimos anos, abordagens baseadas em aprendizado profundo tornaram-se cada vez mais populares, já que em muitos casos estão superando completamente as abordagens clássicas. Dentre os métodos baseados em RNPs, um que vem se destacando, conseguindo desempenho

comparável, ou melhor, que os métodos, do nível do estado da arte existentes para conjuntos de dados complexos, é a rede neural de classe única (do inglês, *one class neural network* – OC-NN) (CHALAPATHY *et al.*, 2018; RUFF *et al.*, 2018).

A OC-NN consiste em treinar um modelo de RNP, usualmente uma rede autocodificadora (GUTOSKI *et al.*, 2017; PINHEIRO *et al.*, 2020; SANTOS *et al.*, 2021), com vários exemplos pertencentes a uma única classe. De modo que, o modelo, através das camadas ocultas, extraia e aprenda os fatores comuns de variação na distribuição dos dados somente dessa classe específica. Após o treinamento, ao ser apresentado a uma entrada que não pertence à classe para a qual este foi treinado, as camadas ocultas do modelo não conseguirão representar essa entrada da forma correta, e essa incapacidade de representar a entrada é mostrada de alguma forma numérica, como uma “pontuação de novidade” (do inglês, *novelty score*). Para entradas conhecidas a pontuação de novidade será baixa, enquanto para entradas desconhecidas a pontuação será alta, definindo-se um valor de limiar para a pontuação, as entradas que apresentarem uma pontuação maior que o limiar serão consideradas observações desconhecidas (CHALAPATHY & CHAWLA, 2019; MARKOU & SINGH, 2003; PIMENTEL *et al.*, 2014).

3.2. DESCRIÇÃO DO SISTEMA

O sistema proposto por este trabalho possui uma estrutura modular, formada por três módulos principais, onde cada módulo é responsável por realizar uma tarefa específica.

Conforme mostrado na Figura 18, o primeiro módulo do sistema é o de detecção de anomalias (MDA), cujo objetivo é monitorar as variáveis de estado da usina nuclear e detectar quando o comportamento dessas variáveis difere do padrão operacional normal. O próximo módulo é o de classificação de acidentes postulados (MCAP), que tem como foco a tarefa de classificar a anomalia previamente detectada, como um dos acidentes de base de projeto postulados da usina. O módulo final do sistema é o de validação (MV), que tem a função de validar se a anomalia identificada, no MCAP, realmente representa o acidente postulado ao qual foi atribuída. Portanto, caso a anomalia identificada não atenda aos critérios de validação do MV, este módulo será responsável por gerar a resposta “Não Sei”, significando que a anomalia detectada é

desconhecida e não pode ser atribuída a nenhum dos acidentes postulados da usina nuclear.

Uma das principais características do sistema proposto é que as tarefas realizadas por cada módulo são executadas através de modelos de RNP. Portanto, cada módulo possui um ou mais modelos de RNP como seu núcleo de operação. Além da exatidão, o uso de RNPs permite ao sistema uma execução rápida, uma vez que, após o seu treinamento, as RNPs podem ser executadas em menos de um segundo em computadores modernos. Além disso, a estrutura modular garante a flexibilidade do sistema na sua execução, por exemplo, o único módulo que está constantemente ativo é de MDA, os demais módulos só são acionados quando é detectada alguma anomalia.

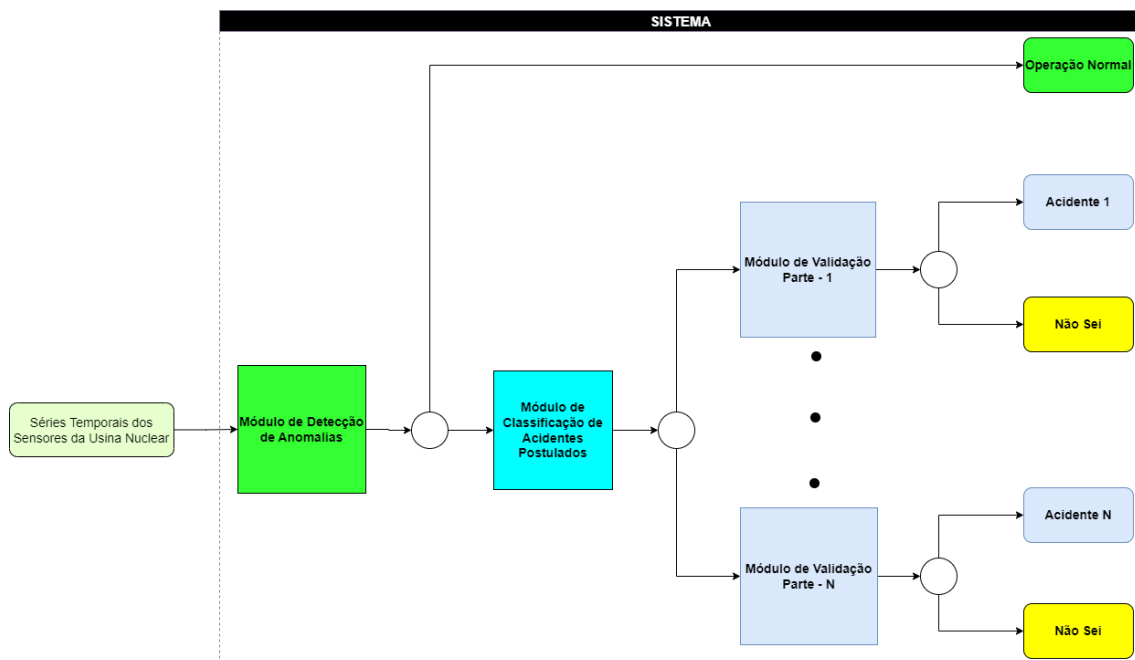


Figura 18. Estrutura do sistema proposto de identificação e diagnóstico de acidentes nucleares (De autoria própria)

3.2.1. Protótipo da Interface do Sistema Proposto

A interface desenvolvida para o sistema, mostrada na Figura 19, possui 2 seções principais. A primeira é a de “Monitoração Estado Operacional” que tem a principal função de mostrar o estado operacional que está sendo identificado pelo sistema. Esta seção é formada pelos seguintes elementos:

- **Campo Data Hora:** Exibe a data hora;

- **Campo Estado Operacional:** Exibe o nome do estado operacional que está sendo identificado pelo sistema ou “Não Sei”. Caso o nome seja “normal, este é exibido em verde, caso um acidente postulado seja identificado o seu nome aparece em vermelho e caso a resposta “Não Sei” seja gerada esse nome aparece em amarelo.
- **Botão Iniciar:** Inicia a monitoração do estado operacional e de variáveis de estado do sistema.
- **Botão Parar:** Para a monitoração do estado operacional e de variáveis de estado do sistema.

A segunda seção é de “Monitoração de Variáveis de Estado”, a qual tem a função de exibir em tempo real os valores das variáveis de estado selecionadas pelo usuário. Esta seção é formada pelos seguintes elementos:

- **Tabela de Monitoração de Variáveis:** Tabela composta pelas colunas:
 - **Variável:** Exibe o nome da variável;
 - **Unidade:** Exibe a unidade da variável;
 - **Valor:** Exibe o valor atual da variável. A cor dos valores muda de acordo a resposta que está sendo exibida pelo sistema no Campo Estado Operacional, sendo verde para estado operacional normal, vermelho para acidente postulado e amarelo para resposta “Não Sei”.
- **Combo Box Adicionar Variável de Estado:** Contém a lista de variáveis disponíveis para o sistema. Ao selecionar uma variável da lista está será adicionada à Tabela de Monitoração de Variáveis.
- **Botão Adicionar Todas as Variáveis de Estado:** Adiciona todas as variáveis disponíveis para o sistema na Tabela de Monitoração de Variáveis.
- **Botão Remover Variável de Estado Selecionada:** Remove uma variável selecionada da Tabela de Monitoração de Variáveis. Para selecionar a variável que se deseja remover é necessário clicar em uma das colunas da linha onde está a variável na tabela.

- **Botão Remove Todas Variáveis de Estado:** Remove todas as variáveis da Tabela de Monitoração de Variáveis.

The figure displays three screenshots of the 'Sistema de Identificação de Acidentes Postulados' interface, illustrating the system's state and the corresponding monitoring data.

Top Left Screenshot (Normal State):

- Operational State:** Data Hora: 12/04/2023 14:04:48, Estado Operacional: **normal**.
- Monitoring Variables Table:**

Sensor	Unidade	Valor
1 Vazao no nucleio	%	105.413
2 Temperatura na perna quente	C	324.293
3 Temperatura na perna fria	C	291.337
4 Vazao no nucleio	kg/s	50840.6
5 Nivel no gerador de vapor - Faixa larga	%	50.1882
6 Nivel no gerador de vapor - Faixa estreita	%	62.5273
7 Pressao no gerador de vapor	Mpa	6.8806
8 Vazao de agua de alimentacao	kg/s	527.989
9 Vazao de vapor	kg/s	528.366
10 Vazao no circuito primario	kg/s	105.414
11 Pressao no sistema primario	Mpa	15.7825
12 Potencia Termica	%	99.8796
13 Potencia Nuclear	%	99.8796
14 Margem de sub-resfriamento	C	21.9603
15 Nivel do pressurizador	%	58.8031
16 Temperatura media no primario	C	307.809

Top Right Screenshot (Blackout State):

- Operational State:** Data Hora: 12/04/2023 14:01:39, Estado Operacional: **blackout**.
- Monitoring Variables Table:**

Sensor	Unidade	Valor
1 Vazao no nucleio	%	19.2906
2 Temperatura na perna quente	C	300.161
3 Temperatura na perna fria	C	295.929
4 Vazao no nucleio	kg/s	47315.3
5 Nivel no gerador de vapor - Faixa larga	%	0.9616
6 Nivel no gerador de vapor - Faixa estreita	%	55.7728
7 Pressao no gerador de vapor	Mpa	8.1666
8 Vazao de agua de alimentacao	kg/s	31.8262
9 Vazao de vapor	kg/s	-4.5084
10 Vazao no circuito primario	kg/s	19.2959
11 Pressao no sistema primario	Mpa	15.359
12 Potencia Termica	%	3.3714
13 Potencia Nuclear	%	3.3714
14 Margem de sub-resfriamento	C	42.812
15 Nivel do pressurizador	%	43.496
16 Temperatura media no primario	C	298.391

Bottom Screenshot (Não Sei State):

- Operational State:** Data Hora: 12/04/2023 13:56:03, Estado Operacional: **Não Sei**.
- Monitoring Variables Table:**

Sensor	Unidade	Valor
1 Vazao no nucleio	%	105.408
2 Temperatura na perna quente	C	324.309
3 Temperatura na perna fria	C	291.358
4 Vazao no nucleio	kg/s	50839.7
5 Nivel no gerador de vapor - Faixa larga	%	50.1614
6 Nivel no gerador de vapor - Faixa estreita	%	62.5273
7 Pressao no gerador de vapor	Mpa	6.8827
8 Vazao de agua de alimentacao	kg/s	528.343
9 Vazao de vapor	kg/s	528.361
10 Vazao no circuito primario	kg/s	105.409
11 Pressao no sistema primario	Mpa	15.7846
12 Potencia Termica	%	99.866
13 Potencia Nuclear	%	99.866
14 Margem de sub-resfriamento	C	21.9561
15 Nivel do pressurizador	%	58.8477
16 Temperatura media no primario	C	307.827

Figura 19. Interface do Sistema em Estado Operacional Normal, em Transiente/Acidente Postulado e em Resposta “Não Sei” (De autoria própria)

3.3. ARQUITETURAS DE REDES NEURAIS PROFUNDAS ANALISADAS PARA O SISTEMA PROPOSTO

3.3.1. Rede Neural Retificada Profunda

A rede neural retificada profunda (do inglês, *deep rectifier neural network* - DRNN), é uma RNP de arquitetura progressiva proposta por Glorot et al., (2011) que tem como principal diferencial o uso de funções retificadas para a ativação dos neurônios ocultos. Como mostrado na seção 2.3.1.1, o principal motivo para o desaparecimento de gradiente em RNPs é uso de funções sigmóides para ativação dos neurônios ocultos, em virtude disso a DRNN de Glorot *et al* (2011) usa uma função retificada, mais especificamente a ReLu.

A ReLu é uma função de ativação matematicamente justificada e biologicamente inspirada, que foi usada pela primeira vez por Hahnloser *et. al*, (2000). Esta função é matematicamente definida como $f(x) = \max(0, x)$, de forma que, quando $x < 0$ a saída é 0, e quando $x \geq 0$ a função produz uma saída linear. Devido a sua definição, na maior parte dos casos, uma RNP que implanta a ReLu não sofre do problema de desaparecimento de gradiente, pois, a ReLu em seu domínio positivo ($x > 0$) apresenta um comportamento linear. Devido a essa linearidade, o gradiente flui bem no caminho dos neurônios ativos durante a retropropagação, uma vez que a derivada na região positiva é sempre 1. A Figura 20 ilustra a função ReLu e sua derivada.

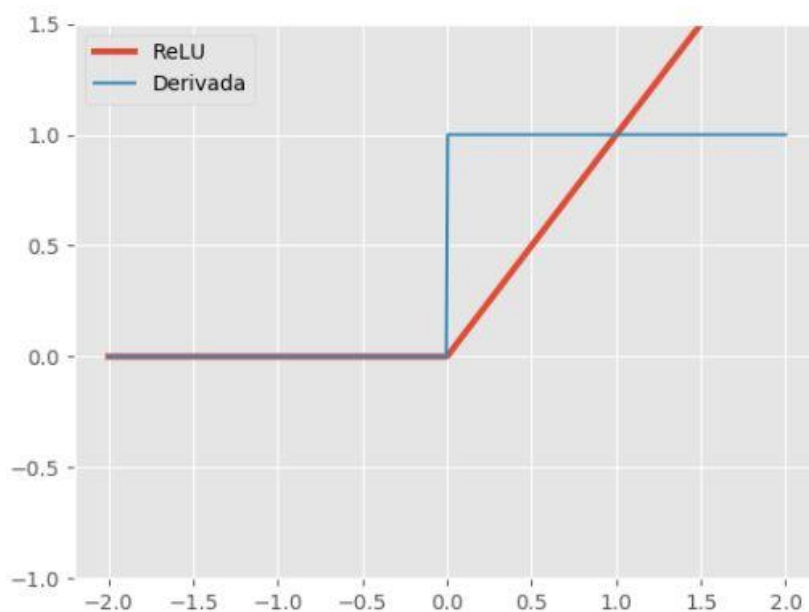


Figura 20. Função ReLu e sua derivada (FACURE, 2017)

Os experimentos de Glorot *et al* (2011) mostraram que as DRNNs apresentam melhor desempenho que outras RNPs progressivas ativadas por função sigmoide. Ademais, estudos recentes na literatura, inclusive na área nuclear, demonstraram que a DRNN é realmente um modelo de RNP progressiva superior, em precisão e tempo de treinamento, em comparação com modelos de RNA progressiva ativados por funções sigmoide (PEDAMONTI, 2018; DESTERRO *et al.*, 2020; SANTOS, *et al.*, 2019a; PINHEIRO *et al.*, 2020; TEIXEIRA *et al.*, 2020; DAM *et al.*, 2021; SALGADO *et al.*, 2021).

3.3.2. Rede Totalmente Convolutacional

A rede totalmente convolutacional, FCN, é um modelo de CNN inicialmente idealizado por Long *et al.*, (2015) para a tarefa de segmentação de imagens, posteriormente, adaptada por Wang *et al.*, (2017) para a tarefa de CST. No contexto da FCN para CST de Wang *et al.*, (2017), esta é um modelo de CNN que possui duas características principais que a distinguem, sendo a primeira a remoção das camadas de subamostragem que procedem às camadas convolucionais, ou seja, não há camadas de subamostragem entre as camadas convolucionais. Segundo os autores, isso foi feito com o intuito de fazer com que o modelo fosse mais resistente ao sobreajuste (*overfitting*). Já a segunda principal característica desse modelo de FCN, é a substituição da camada totalmente conectada, que precede a camada de saída, por uma camada de subamostragem média global (do inglês, *global average pooling layer* - GAP) (ZHOU *et al.*, 2016).

Sobre a camada de GAP, esta realiza um tipo de operação subamostragem, onde uma grade-3D (x, y, z) de mapas de características, sendo x, y e z , respectivamente, o comprimento, altura dos mapas e a quantidade de mapas, é reduzida para uma grade-3D ($1, 1, z$) através do cálculo da média de todos os elementos de cada mapa. O processo da GAP para uma grade-2D (x, z) é o mesmo, reduzindo a grade-2D ($1, z$), com número de mapas de características sendo preservado. A principal vantagem da utilização da GAP, no lugar da camada totalmente conectada, a redução no número de parâmetros de aprendizado da rede, agilizando o processo de treinamento, mas preservando a capacidade de generalização (YAMASHITA *et al.*, 2018).

Retornando à FCN de Wang *et al.*, (2017), esta possui três camadas convolucionais, com a primeira camada tendo 128 filtros 1D de tamanho 8; a segunda

com 256 filtros 1D de tamanho 5; e a terceira, possuindo 128 filtros 1D de tamanho 3. Em todas as 3 camadas, a operação de convolução é realizada com valor stride = 1, utilizando o preenchimento de zeros, quando necessário, para preservar o tamanho exato da série temporal após as convoluções. Ao final do processo de extração de características, o tamanho dos 128 mapas de características resultantes da terceira camada de convolucional, é reduzido para um pela camada GAP, resultando em uma grade de tamanho 1x128 que se conecta com a camada de saída.

Nos experimentos conduzidos por Wang *et al.*, (2017), utilizando 44 conjuntos de dados de problemas de CST para séries temporais univariáveis (YANPING *et al.*, 2015), a FCN alcançou o melhor desempenho, superando outras abordagens clássicas e modernas (de aprendizado profundo). Recentemente, no trabalho de Fawaz *et al.*, (2019) a capacidade da FCN em lidar com problemas de CST foi novamente confirmada, ao ser um dos modelos de RNP a alcançar os melhores resultados, em um experimento comparativo contendo 96 conjunto de dados de problemas de CST para séries temporais, univariáveis quanto multivariável.

3.3.3. Long Short-Term Memory

A *Long Short-Term Memory* (LSTM) (HOCHREITER & SCHMIDHUBER, 1997) é uma RNA de arquitetura recorrente que busca melhorar o fluxo do gradiente durante o treinamento, permitindo o aprendizado de longas sequencias. Sendo assim, a LSTM possui as mesmas características de uma rede recorrente tradicional, entretanto, a principal diferença da LSTM está na utilização de uma “memória” de longo prazo, chamada estado da célula, que tem a função de, durante o treinamento, preservar as informações mais relevantes para o aprendizado de uma sequência, assim como descartar as informações não relevantes.

Em uma rede LSTM, os neurônios das camadas ocultas são chamados de blocos de memória ou células. A célula de uma LSTM, como mostra a Figura 21, diferentemente das RNNs que possuem apenas duas entradas (estado interno anterior e o elemento de entrada atual), a célula LSTM possui uma entrada adicional que é o estado interno (memória de longo prazo). O controle do fluxo das informações que entram ou são removidas do estado da célula é feito através de três unidades multiplicativas, chamadas de portões. Sendo assim, usando a Figura 21 como referência, os portões controlam o fluxo de informação na célula LSTM da seguinte forma:

- **Etapa 1 – Portão de Esquecimento:** Decide quais informações, do estado da célula anterior (c_{t-1}) devem ser descartadas ou mantidas. Este portão recebe como entradas as informações do estado interno anterior (h_{t-1}) e o elemento de entrada atual (x_t), concatena essas entradas ($[h_{t-1}, x_t]$), em seguida passando por uma função sigmoide logística: $f_t = \sigma(W_f[h_{t-1}, x_t])$. Um valor resultante próximo de 1 sinaliza que a informação deve ser mantida no estado da célula. Enquanto um valor resultante próximo de 0, sinaliza que a informação deve ser esquecida.
- **Etapa 2 – Portão de Entrada:** Determina a nova informação a ser armazenada no estado da célula. Este portão é dividido em duas etapas:
 - **Etapa 2.1 - Sigmoide:** $[h_{t-1}, x_t]$ passam por uma função sigmoide logística para decidir quais valores serão atualizados: $i_t = \sigma(W_i[h_{t-1}, x_t])$.
 - **Etapa 2.2 - Tanh:** $[h_{t-1}, x_t]$ passam por uma função tangente hiperbólica para gerar um vetor com possíveis valores a adicionar ao estado da célula: $\tilde{c}_t = \tanh(W_c[h_{t-1}, x_t])$
- **Etapa 3 – Atualização do Estado da Célula:** Com as informações geradas pelos portões de esquecimento e o de entrada, o estado da célula (c_t) pode ser atualizado. Sendo assim, é aplicada a operação de esquecimento ao estado da célula anterior, e são adicionados os possíveis valores, escalados pelo quanto se decidiu atualizar: $c_t = f_t * c_{t-1} + i_t * \tilde{c}_t$.
- **Etapa 4 – Portão de Saída:** Tem a função de decidir qual deve ser o próximo estado interno (h_t). Para isso, primeiro $[h_{t-1}, x_t]$ passam por uma função sigmoide: $o_t = \sigma(W_o[h_{t-1}, x_t])$. Em seguida, o estado de célula recém atualizado passa por uma a função tangente hiperbólica: $\tanh(c_t)$. O resultado das duas funções é então multiplicado para decidir quais informações o estado interno deve transportar: $h_t = o_t * \tanh(c_t)$.

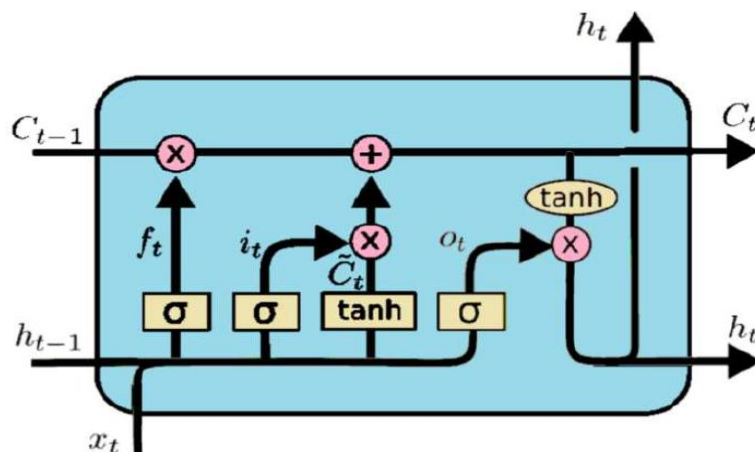


Figura 21. Célula LSTM (OLAH, 2015)

Analisando as etapas apresentadas do fluxo de informação em uma célula LSTM, é possível observar que há apenas multiplicações elemento a elemento (produtos de Hadamard), combinadas por uma soma. Sendo assim, ao retropropagar o erro na LSTM, o fluxo do gradiente não tende a ser interrompido.

3.3.4. Long Shot-Term Memory-Fully Convolution Network

A *Long Shot-Term Memory-Fully Convolution Network* (LSTM-FCN) é um modelo de RNP híbrido, que une estruturas de duas arquiteturas de RNA, no caso a recorrente e convolucional, desenvolvido por Karim *et al.*, (2017) que tenta melhorar o desempenho do modelo de FCN de Wang *et al.*, (2017 em tarefas de CST, adicionado um submódulo LSTM. Uma das principais vantagens do modelo LSTM-FCN é que as séries temporais de entrada não passem por nenhum tipo de pré-processamento intenso.

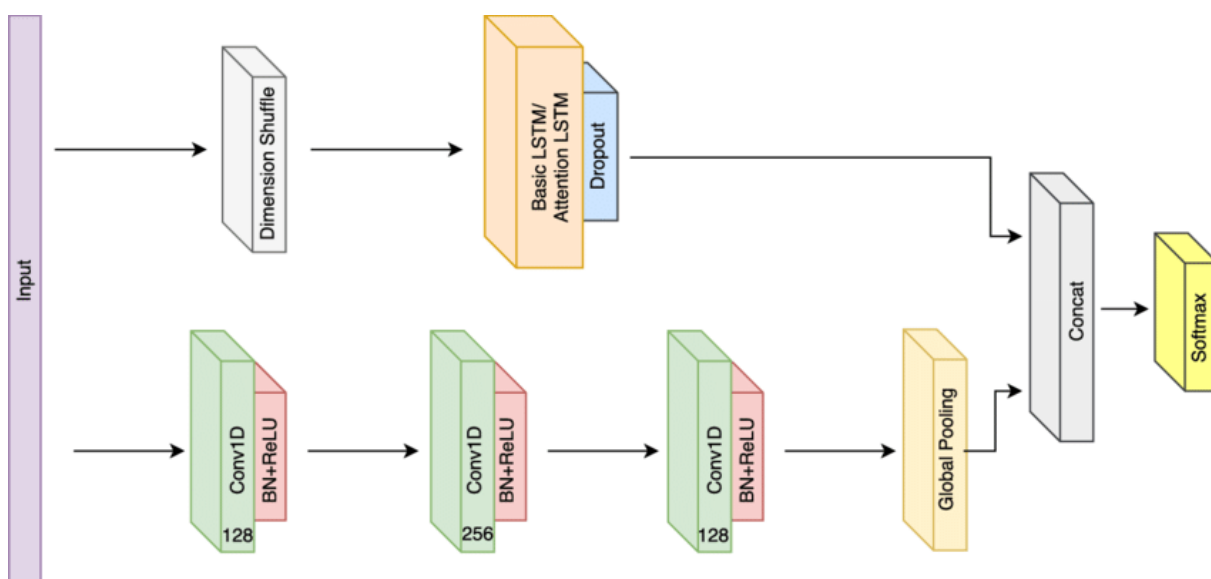


Figura 22. Estrutura interna da LSTM-FCN (KARIM *et al.*, 2017).

Como mostra a Figura 22, a LSTM-FCN divide a sua estrutura interna em dois caminhos paralelos, um formado pelas estruturas da FCN e outro pela LSTM, Karim *et al.*, (2017) nomeiam esses caminhos, como bloco FCN e bloco LSTM. Ao receber uma série temporal de entrada, esta percorre os dois blocos simultaneamente, contudo, o modo como cada bloco interpreta essa entrada é diferente. O bloco FCN interpreta a entrada como uma série temporal univariada com vários passos temporais. Se houver uma série temporal de comprimento N, o bloco convolucional receberá os dados em N passos de tempo. Por outro lado, o bloco LSTM recebe a entrada como uma série temporal multivariada, com um único passo temporal. Essa maneira de interpretar a entrada é proporcionada via camada de embaralhamento de dimensão (do inglês, *dimension shuffle*) que transpõe a dimensão temporal da série temporal. Ou seja, uma série temporal univariada de comprimento N, após a transposição, será vista como uma série temporal multivariada (com N variáveis), com um único passo no tempo.

Segundo Karim et al. (2017), a forma como o bloco LSTM interpreta a entrada é essencial para o desempenho aprimorado da LSTM-FCN. Quando essa forma não é utilizada, o desempenho da LSTM é reduzido significativamente, de modo que em conjuntos de dados formados por séries temporais curtas, a LSTM-FCN tende a sobreajustar, enquanto em conjuntos de dados com séries temporais longas, a LSTM-FCN tem dificuldade em aprender as correlações temporais da série temporal.

Ademais, a camada LSTM pode ser formada por células LSTM tradicionais ou por células de LSTM com mecanismo de atenção (BAHDANAU *et al.*, 2015). A versão da LSTM-FCN que usa células de LSTM com mecanismo de atenção é chamada ALSTM-FCN. Ambas as versões, LSTM-FCN e ALSTM-FC, foram testadas com 85 conjuntos de dados com diferentes problemas de CST, formados por séries temporais univariadas. O resultado obtido pelos dois modelos foi considerado uma melhoria no estado da arte para classificação de séries temporais usando RNP. Além disso, entre a LSTM-FCN e a ALSTM-FCN, a versão mais simples (LSTM-FCN) obteve resultados superiores.

3.2.4.1. Multivariate Long Shot-Term Memory-Fully Convolution Network

Embora a LSTM-FCN tenha demonstrado bom desempenho na tarefa CST, está só havia sido testada para trabalhar com séries temporais univariadas, em virtude disso, Karim *et al.*, (2019) elaborou um novo modelo de LSTM-FCN para trabalhar com séries

temporais multivariadas, denominado *Multivariate Long Shot-Term Memory-Fully Convolution Network* (MLSTM-FCN).

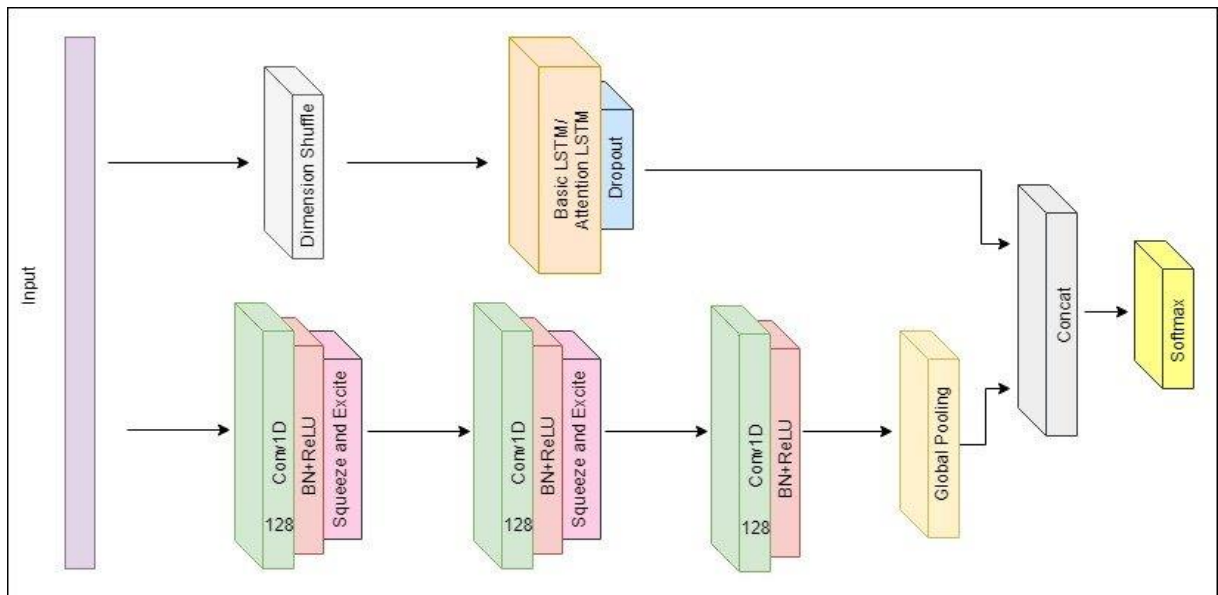


Figura 23. Estrutura interna da Multivariate LSTM-FCN (KARIM *et al.*, 2019)

A estrutura interna da MLSTM-FCN, mostrada na Figura 23, é similar à da LSTM-FCN original, a principal diferença estrutural se encontra no bloco FCN, com a introdução de uma camada de compressão-e-excitação (do inglês, *Squeeze-and-Excitation - SE*) (HU *et al.*, 2018). A SE visa aperfeiçoar a qualidade das representações de uma camada convolucional, modelando explicitamente as interdependências entre os canais dos mapas de características. Para isso, a SE usa operações de compressão e excitação para executar uma recalibração de características, fazendo com que a rede aprenda a usar informações globais, para enfatizar seletivamente as características mais relevantes e informativas da entrada e suprimir as características menos úteis.

Em relação ao fluxo da entrada na MLSTM-FCN, como os conjuntos de dados agora consistem em séries temporais multivariadas, as entradas têm o formato de uma grade-3D (N, Q, M) , em que N é o número de exemplos no conjunto de dados, Q é o número máximo de passos de todas as variáveis e M é o número de variáveis. No caso de um conjunto de dados com séries temporais univariadas, o valor de M seria 1.

Sendo assim, a entrada para o bloco FCN é interpretada como uma série temporal multivariada com Q passos temporais, e com M variáveis distintas. Já para o bloco LSTM, se a camada de embaralhamento de dimensão não for aplicada, a camada LSTM exigirá Q etapas de tempo para processar as variáveis M em cada passo de

tempo. No entanto, se o embaralhamento de dimensão for aplicado, a camada LSTM exigirá M passos de tempo para processar Q variáveis por passo de tempo. Em outras palavras, o embaralhamento de dimensão melhora a eficiência do modelo quando o número de variáveis M é menor que o número de passos Q . Portanto, após o embaralhamento de dimensão, em cada passo t , em que $1 \leq t \leq M$, sendo M o número de variáveis, a entrada recebida pela camada LSTM fornece todo o histórico dessa variável (dados dessa variável em todos os passos de tempo Q). Deste modo, a camada LSTM recebe informações temporais globais de cada variável de uma única vez. Como resultado, a operação embaralhamento de dimensão reduz o tempo de cálculo do treinamento e da inferência da rede, sem perder a exatidão para problemas de CST (KARIM *et al.*, 2019).

A MLSTM-FCN ao ser avaliada em 35 conjuntos de dados de séries temporais multivariável, atingiu resultado do nível do estado da arte em 28 deles. Segundo Karim *et al.*, (2019), a MLSTM-FCN pode ser aplicada em várias tarefas de CST com séries temporais multivariadas, podendo ser implantada rapidamente em sistemas de tempo real e sistemas embarcados, dado que o modelo de MLSTM-FCN é pequeno e eficiente.

3.3.5. LSTM Autoencoder

O *Long Shot-Term Memory* (LSTM-AE) é um modelo de autoencoder de arquitetura recorrente, que consegue modelar e aprender a reproduzir dados de entrada sequenciais, como séries temporais. Originalmente, o LSTM-AE foi projetado por Sutskever *et al.*, (2014) para a área de processamento de linguagem natural, mais especificamente para a tarefa de tradução automática. Posteriormente, o LSTM-AE foi adaptado para a reconstrução de sequências de valores contínuos, como séries temporais.

A arquitetura do LSTM-AE, assim como o AE clássico mostrado no capítulo “2.4.4. Redes Neurais Autocodificadoras”, também é formada por uma estrutura codificadora e decodificadora, contudo, no LSTM-AE estas duas estruturas são formadas a partir de camadas LSTM. Como mostra a Figura 24, a camada LSTM codificadora recebe uma sequência de entrada (x_1, x_2, \dots, x_n) e a codifica em um vetor de representação de tamanho fixo. Em seguida, a camada LSTM decodificadora recebe o vetor como entrada e tenta reconstruir a sequência de entrada em (x_1, x_2, \dots, x_n) . Sendo assim, o objetivo do LSTM-AE é a partir do seu processo de aprendizado,

reduzir o erro de reconstrução, de modo a aprender a reconstruir a sequência de entrada como a sequência de saída.

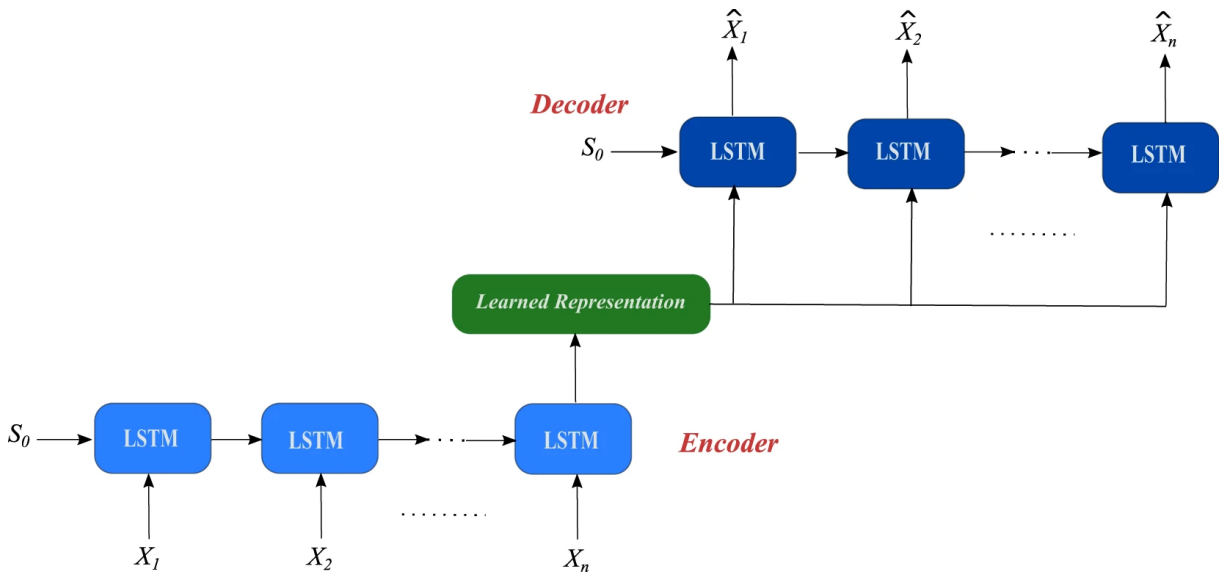


Figura 24. Estrutura do LSTM-Autoencoder (SAGHEER & KOTB, 2019)

3.4. MATERIAIS E MÉTODOS

O tempo de treinamento é um aspecto relevante para arquiteturas profundas de redes neurais; no passado, este era um dos problemas que limitavam o uso destas arquiteturas. Em virtude disso, atualmente, a computação paralela baseada em Unidade de Processamento Gráfico (em inglês *Graphics Processing Unit* - GPU) é o método mais usado para acelerar o treinamento dos modelos de aprendizagem profunda. Nesse contexto, neste trabalho, o TensorFlow (ABADI *et al.*, 2016) foi utilizado para o desenvolvimento e o treinamento desses modelos.

O TensorFlow é um *framework* de aprendizagem profunda de código aberto desenvolvida pela equipe de aprendizado de máquina do Google, que oferece uma biblioteca de software que facilita o desenvolvimento, o treinamento e validação de redes neurais profundas por meio de linguagens de programação de alto nível (como C++ ou Python). Em relação ao hardware, a GPU utilizada foi a GeForce GTX 1070. Esta GPU implanta a arquitetura Pascal, que é a sexta geração de dispositivos NVIDIA capazes de usar a tecnologia *Compute Unified Device Architecture* (CUDA). As GPUs habilitadas para CUDA permitem que os vários núcleos de GPU, usados anteriormente apenas em aplicativos gráficos, sejam usados para executar tarefas de aplicações de uso geral, agilizando assim o processamento.

CAPÍTULO 4

ESTUDO DE CASO, EXPERIMENTOS E RESULTADOS

4.1. APRESENTAÇÃO DO ESTUDO DE CASO

4.1.1. Descrição do Conjunto de Dados

Nesta tese, como estudo de caso, foi utilizado um conjunto de dados gerado por Alvarenga (1997) a partir de simulações de estados operacionais para a usina nuclear Angra 2 para treinar, validar e testar os modelos de RNP que compõem os módulos do sistema proposto. No total, o conjunto de dados é formado por 16 estados operacionais, a operação normal e 15 acidentes postulados, como mostra a Tabela 1. Cada um desses estados é representado pela evolução temporal de 61 segundos de 16 variáveis de estado. Na Tabela 2, são listadas as variáveis de estado que caracterizam cada um dos estados operacionais. Sendo assim, a partir dessa descrição, o conjunto de dados utilizado neste estudo de caso, pode ser definido como um conjunto de 16 séries temporais multivariadas (estados operacionais), onde cada uma dessas séries multivariadas é composta por 16 séries temporais únicas de tamanho 61 (variáveis de estado), ou seja, 61 observações no tempo.

Tabela 1. Estados Operacionais

ID	Estado	Descrição
1	BLACKOUT	Perda de alimentação elétrica externa
2	BLACKSEM	Perda de alimentação elétrica sem desligamento do reator
3	LOCA	Perda de refrigerante do sistema primário
4	MEFWISEM	Isolamento da alimentação principal e auxiliar sem desligamento do reator
5	MEFWISO	Isolamento da alimentação principal e auxiliar
6	MFWBR	Ruptura da alimentação principal
7	MFWBRSEM	Ruptura da alimentação principal sem desligamento do reator
8	MFWISEM	Isolamento da alimentação principal sem desligamento do reator
9	MFWISO	Isolamento da alimentação principal
10	MSTMISEM	Isolamento da linha de vapor principal sem desligamento do reator
11	MSTMISO	Isolamento da linha de vapor principal
12	NORMAL	Condição normal de potência
13	SGTR	Ruptura de tubos do gerador de vapor
14	STMLIBR	Ruptura da linha de vapor principal
15	TRIPREA	Desligamento da turbina sem desligamento do reator
16	TRIP	Desligamento da turbina e do reator

Tabela 2. Variáveis de Estado

ID	Variável	Unidade
1	Vazão no núcleo	%
2	Temperatura na perna quente	°C
3	Temperatura na perna fria	°C
4	Vazão no núcleo	kg/s
5	Nível no gerador de vapor – faixa larga	%
6	Nível no gerador de vapor – faixa estreita	%
7	Pressão no gerador de vapor	MPa
8	Vazão de água de alimentação	kg/s
9	Vazão de vapor	kg/s
10	Vazão no circuito primário	kg/s
11	Pressão no sistema primário	MPa
12	Potência térmica	%
13	Potência nuclear	%
14	Margem de sub-resfriamento	°C
15	Nível do pressurizador	%
16	Temperatura média no primário	°C

4.1.2. O Método de Aumento de Dados

No contexto das RNAs, especialmente em RNPs, a quantidade de exemplos utilizados para treinar os modelos é um aspecto de grande importância. Uma vez que o desempenho superior das RNPs depende da quantidade de exemplos disponível para o treinamento, de modo a evitar o sobreajuste (do inglês, *overfitting*), que é quando o modelo de RNP se ajusta exatamente bem ao conjunto de treinamento, mas não consegue generalizar para novos dados. Em virtude disso, uma estratégia bastante implantada ao trabalhar com modelos de aprendizagem profunda é o uso de métodos de “aumento de dados” (do inglês, *data augmentation* - DA), que consiste em usar métodos que aumentem significativamente a diversidade de dados disponíveis para o treinamento dos modelos, sem realmente coletar novos dados.

O uso de DA é vastamente difundido no campo do processamento de imagens com RNPs, com vários métodos disponíveis (PEREZ & WANG, 2017). Em comparação, no campo do processamento de séries temporais com RNPs, a elaboração e aplicação de métodos de DA é uma prática menos difundida. Contudo, quando uma estratégia de DA é aplicada em séries temporais, o desempenho de modelos de RNPs melhora significativamente (WEN *et al.*, 2020).

Sendo assim, visando melhorar a robustez do método proposto, e, em simultâneo, fornecer aos modelos de aprendizagem profunda uma quantidade adequada

de exemplos de treinamento, neste trabalho é proposta a utilização do ruído de instrumentação, ou seja, as incertezas dos instrumentos de medição (tipicamente consideradas 1% nas usinas nucleares), como o método de DA. Para aplicar o método de DA, cada variável, de cada uma das séries temporais multivariadas, foi sobreposta com um ruído branco com distribuição normal e 1% de desvio padrão.

4.1.3. A Técnica de Janela Deslizante

A técnica de janela deslizante (*sliding window* - SW) é um método utilizado para facilitar a modelagem e análise de uma série temporal. A técnica de SW consiste em utilizar uma janela temporal de tamanho fixo para dividir uma sequência temporal em múltiplas subsequências que possuem o mesmo tamanho da janela utilizada, de modo que, considerando uma série temporal T de tamanho n e uma janela temporal de tamanho w , uma matriz D é construída ao deslizar a janela ao longo de T , colocando a subsequência S_p na p -ésima linha de D , sendo o tamanho de $D = [(n - w + 1), w]$ (BATISTA, 2010).

Visando exemplificar a técnica de SW. Na Figura 25, é mostrada a aplicação da técnica, utilizando uma janela temporal de tamanho $w=5$, em uma série temporal de tamanho $n=10$. Como resultado, são geradas 6 sequências que formam a matriz $D[6, 5]$.

A técnica de SW é amplamente utilizada para modelar séries temporais utilizando redes convolucionais e recorrentes. Sendo assim, nesse trabalho, para os modelos de RNP de arquitetura recorrente e convolucional, também é utilizada a técnica de SW para modelar o conjunto de dados de séries temporais de acidentes nucleares.

Série Temporal Original	1	2	3	4	5	6	7	8	9	10
1º Subsequência	1	2	3	4	5					
2º Subsequência		2	3	4	5	6				
3º Subsequência			3	4	5	6	7			
4º Subsequência				4	5	6	7	8		
5º Subsequência					5	6	7	8	9	
6º Subsequência						6	7	8	9	10

Tamanho Série Temporal (n) = 10
Tamanho Janela Temporal (w) = 5
Matriz D = [(10 - 5 + 1), 5]
Matriz D = [6, 5]

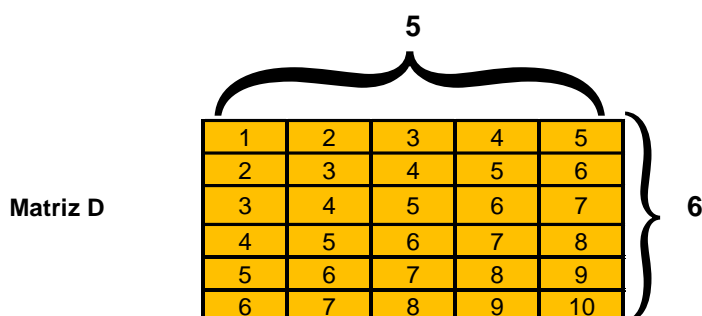


Figura 25. Exemplo da técnica de janela deslizante (De autoria própria)

4.1.4. Modelagem e Distribuição do Conjunto de Dados

Originalmente, cada classe (estado operacional) do conjunto de dados está representada por uma série temporal multivariada com tamanho 61, o que significa que cada série temporal univariada (variáveis de estado) possui 61 observações. Porém, percebeu-se que os eventos realmente começavam na terceira observação, uma vez que as duas primeiras observações ainda são da operação normal. Portanto, a primeira etapa no processo de modelagem de dados foi a remoção das duas primeiras observações de cada série temporal multivariada.

Em seguida, a segunda etapa do processo de modelagem foi a aplicação do método de aumento de dados utilizando ruído instrumental. Sendo assim, para cada estado operacional, foram geradas 299 novas séries temporais multivariadas. No final desta segunda etapa de modelagem, cada classe do conjunto de dados ficou sendo composta de 300 séries temporais multivariadas (a série original, mais 299 séries com ruído) de tamanho 59.

A terceira etapa de modelagem foi a implantação da técnica de SW no conjunto de dados aumentado. Nesse processo, janelas temporais de diferentes tamanhos foram aplicadas ao conjunto de dados aumentado. Como resultado, para cada tamanho de

janela temporal aplicado, uma versão modificada do conjunto de dados aumentado foi gerada.

A Tabela 3 apresenta como o conjunto de dados original foi sendo modificado em cada etapa do processo de modelagem. A primeira linha da Tabela 3 mostra as características do conjunto de dados original após a remoção das duas primeiras observações de cada série temporal multivariada. A segunda linha apresenta as características do conjunto de dados após o aumento de dados. Nesta linha, o valor de 17700 amostras/exemplos por classe é obtido ao multiplicar o número de séries temporais multivariadas em cada classe, que é 300, pelo tamanho das séries temporais, que é 59. Da mesma forma, o valor de 283200 para o número total de amostras é obtido, multiplicando o número de classes (16), pelo número de amostras em cada classe (17700).

Por fim, as linhas restantes da Tabela 3 mostram as características do conjunto de dados aumentado após a implantação da técnica de SW utilizando janelas temporais de diferentes tamanhos. O tamanho das janelas temporais é apresentado na coluna “*Timesteps*”, este nome *timesteps* é utilizado porque, geralmente, quando a técnica de SW é aplicada no contexto de RNAs, o tamanho da janela temporal é denominado *timestep*. Outra característica relevante a mencionar, é que ao utilizar a técnica de SW no contexto de RNAs, o conjunto de dados assume um formato 3D [número de exemplos, *timesteps*, número de variáveis].

Tabela 3. Formatação e Características dos Conjuntos de Dados

Conjunto de Dados	Timesteps	Nº de Classes	Nº de Exemplos por Classe	Nº Total de Exemplos	Nº de Variáveis	Formato do Conjunto de Dados
Conjunto Original	-	16	59	944	16	[944, 16]
Conjunto Aumentado	-	16	17700	283200	16	[283200, 16]
Conjunto Janela Temporal – 5	5	16	16500	264000	16	[264000, 5, 16]
Conjunto Janela Temporal – 10	10	16	15000	240000	16	[240000, 10, 16]
Conjunto Janela Temporal – 15	15	16	13500	216000	16	[216000, 15, 16]
Conjunto Janela Temporal – 20	20	16	12000	192000	16	[192000, 20, 16]
Conjunto Janela Temporal – 25	25	16	10500	168000	16	[168000, 25, 16]

Após o processo de modelagem, os conjuntos de dados que seriam usados nos experimentos com os modelos de RNP foram subdivididos em três subconjuntos:

- **Conjunto de Treinamento:** Pode-se dizer que este é o conjunto mais importante, pois contém os exemplos para treinar a RNA, ou seja, o modelo aprende a representação do problema a partir dos dados presentes neste conjunto.
- **Conjunto de Validação:** Este conjunto é utilizado para avaliar a capacidade de generalização de um modelo de RNA durante o processo de treinamento. Geralmente, esse conjunto é utilizado para verificar se está havendo sobreajuste durante o treinamento da RNA e, também guiar o desenvolvedor no ajuste da configuração de hiperparâmetros do modelo.
- **Conjunto de Teste:** Fornece a avaliação final do desempenho do modelo, uma vez que contém dados novos, que o modelo nunca viu durante o seu treinamento, replicando assim o cenário a que o modelo será testado na etapa de produção.

Neste contexto, a Tabela 4 apresenta as características e distribuição das amostras para cada conjunto de dados.

Tabela 4. Características e Distribuição dos Conjuntos de Dados em Treinamento, Validação e Teste

Conjunto de Dados	Subconjunto	Timesteps	Nº de Classes	Nº de Exemplos por Classe	Nº Total de Exemplos	Nº de Variáveis	Formato do Conjunto de Dados
Conjunto Aumentado	Treinamento	-	16	5900	94400	16	[94400, 16]
	Validação	-	16	5900	94400	16	[94400, 16]
	Teste	-	16	5900	94400	16	[94400, 16]
Conjunto Janela Temporal – 5	Treinamento	5	16	5500	88000	16	[88000, 5, 16]
	Validação	5	16	5500	88000	16	[88000, 5, 16]
	Teste	5	16	5500	88000	16	[88000, 5, 16]
Conjunto Janela Temporal – 10	Treinamento	10	16	5000	80000	16	[80000, 10, 16]
	Validação	10	16	5000	80000	16	[80000, 10, 16]
	Teste	10	16	5000	80000	16	[80000, 10, 16]
Conjunto Janela Temporal – 15	Treinamento	15	16	4500	72000	16	[72000, 15, 16]
	Validação	15	16	4500	72000	16	[72000, 15, 16]
	Teste	15	16	4500	72000	16	[72000, 15, 16]
Conjunto Janela Temporal – 20	Treinamento	20	16	4000	64000	16	[64000, 20, 16]
	Validação	20	16	4000	64000	16	[64000, 20, 16]
	Teste	20	16	4000	64000	16	[64000, 20, 16]
Conjunto Janela Temporal – 25	Treinamento	25	16	3500	56000	16	[56000, 25, 16]
	Validação	25	16	3500	56000	16	[56000, 25, 16]
	Teste	25	16	3500	56000	16	[56000, 25, 16]

4.2. EXPERIMENTOS PARA O MÓDULO DE CLASSIFICAÇÃO DE ACIDENTES POSTULADOS

O desempenho de quatro modelos de RNP foi analisado na tarefa de classificação de acidentes postulados que deve ser cumprida pelo MCAP. Os quatro modelos avaliados foram:

- **DRNN:** Embora seja uma rede progressiva, esta mostrou ótimos resultados em trabalhos anteriores (SANTOS, *et al.*, 2019a e PINHEIRO *et al.*, 2020), onde situações similares à que estão sendo apresentadas neste trabalho foram analisadas. Por isso, a mesma configuração de DRNN avaliada em Santos *et al.*, (2019a) e Pinheiro *et al.*, (2020) foi testada também nos estados operacionais propostos nesse trabalho.
- **FCN:** A arquitetura convolucional desenvolvida por Wang *et al.*, (2017), atualmente é considerada um dos melhores métodos de classificação de séries temporais. Contudo, esta ainda não foi utilizada especificamente no problema de classificação de acidentes nucleares. Em virtude disso, nesse trabalho, a FCN foi avaliada pela primeira vez na classificação de acidentes nucleares.

- **LSTM:** Diversos trabalhos na literatura mostram a capacidade da LSTM em problemas de previsão de séries temporais. Contudo, LSTMs são raramente aplicadas em problemas de classificação de séries temporais. Ainda assim, nos trabalhos da área nuclear, que aplicaram LSTMs para a classificação de acidentes nucleares (SAEED *et al.*, 2020; YANG *et al.*, 2020), bons resultados foram encontrados. Em virtude disso, uma LSTM foi avaliada no cenário proposto.
- **LSTM-FCN:** A arquitetura híbrida, inspirada na FCN de Wang *et al.*, (2017), desenvolvida por Karim *et al.*, (2017), também foi avaliada neste trabalho. Uma vez que a classificação de acidente nucleares é um problema de classificação de séries temporais multivariáveis, a versão mais recente da LSTM-FCN, a MLSTM-FCN (KARIM *et al.*, 2019) desenvolvida para lidar com séries temporais multivariáveis, foi a versão da LSTM-FCN testada nesse trabalho. Dentre as arquiteturas de RNP testadas, a MLSTM-FCN é a mais recente. Nos testes realizados por Karim *et al.*, (2019) o desempenho de MLSTM-FCN na classificação de séries temporais é comparável ao da FCN de Wang *et al.*, (2017).

Deste modo, para os experimentos, todos os modelos de RNP foram treinados, validados e testados com o conjunto de dados apresentado na seção 4.1. Considerando que, dentre os modelos, somente para DRNN a técnica de janela deslizante não foi utilizada para modelar o conjunto de dados. Para os demais modelos de RNP, a janela deslizante foi utilizada e o desempenho dessas RNPs foi avaliado considerando diferentes *timesteps*.

4.2.1. Resultados

A Tabela 5 mostra a exatidão obtida por cada um dos modelos de RNP para o conjunto de teste. Analisando o resultado, é possível perceber que a DRNN, que é a arquitetura progressiva, e não utiliza a técnica de janela deslizante, foi a que obteve o pior resultado. Contudo, para os demais modelos de RNP, à medida que o número de *timesteps* foi aumentando, o desempenho das RNPs foi melhorando. A Figura 26 destaca graficamente essa relação do número de *timesteps* com o desempenho dos modelos.

Embora os modelos LSTM, FCN e LSTM-FCN tenham obtido um comportamento similar, alcançando uma exatidão de 100% com apenas 25 *timesteps*, ao analisar a exatidão média, na Tabela 5, é possível observar que a FCN e LSTM-FCN superaram a LSTM, com a FCN alcançando a maior exatidão média. Esses resultados estão em linha com os da literatura de classificação de séries temporais, onde a FCN vem se destacando, alcançando os melhores resultados em diferentes problemas de classificação de séries temporais (KARIM *et al.*, 2019). De modo que, no problema de classificação de acidentes nucleares isso não foi diferente.

Tabela 5. Exatidão Alcançada pelas RNPs no Conjunto de Teste

TIMESTEPS	DRNN	LSTM	FCN	LSTM-FCN
1	94,18%	-	-	-
5	-	95,29%	96,16%	96,00%
10	-	96,70%	97,68%	97,59%
15	-	98,10%	98,76%	98,86%
20	-	99,22%	99,62%	99,57%
25	-	100%	100%	100%
30	-	100%	100%	100%
35	-	100%	100%	100%
40	-	100%	100%	100%
45	-	100%	100%	100%
50	-	100%	100%	100%
55	-	100%	100%	100%
60	-	100%	100%	100%
Exatidão Média	-	99,10%	99,35%	99,33%

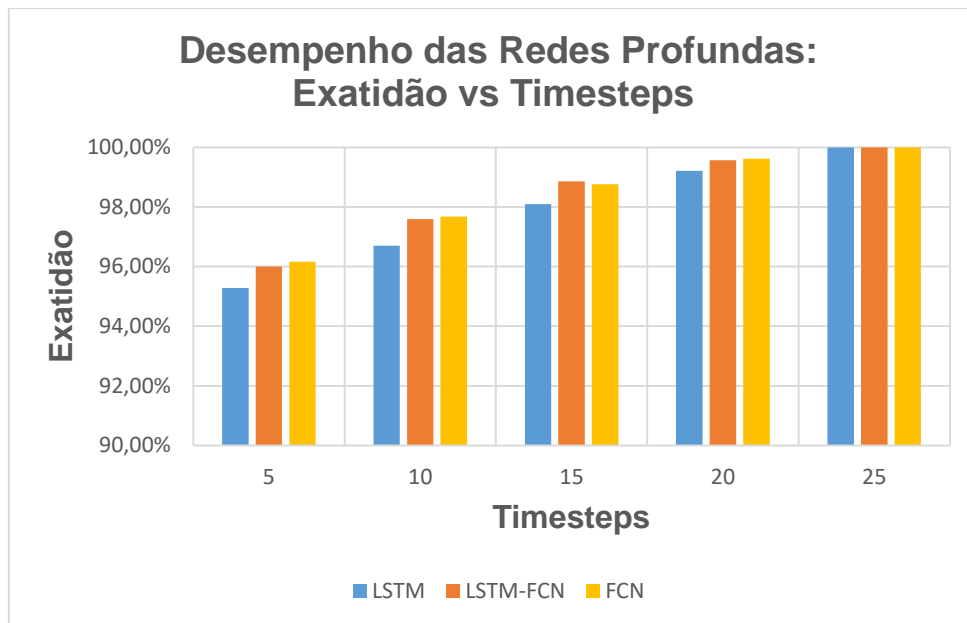


Figura 26. Evolução do Desempenho da LSTM, LSTM-FCN e FCN em relação ao número de *timesteps* (De autoria própria)

De modo a avaliar em mais detalhe o desempenho da FCN matrizes de confusão foram utilizadas. Sendo a matriz de confusão uma tabela que mostra o desempenho de um modelo de classificação, comparando as previsões feitas pelo modelo com as classes reais dos dados. As Figuras 27, 28, 29, 30 e 31, mostram a matriz de confusão gerada pela FCN para diferentes números de *timesteps*. Analisando as matrizes de confusão, a partir da Figura 27, é possível observar que a maior dificuldade da FCN está na classificação correta dos acidentes MEFWISEM, MEFWISO, MFWBR, MFWBRSEM, MFWISEM e MFWISO. Este é um resultado esperado, dado que alguns desses acidentes apresentam um comportamento extremamente parecido. Contudo, à medida que o número de *timesteps* vai aumentando, a quantidade de informação temporal disponível para FCN extrair de cada exemplo também aumenta, de modo que, com 25 *timesteps*, a FCN já não classifica de forma errada esses acidentes e alcança uma exatidão de 100% na classificação de todos os estados operacionais.

Considerando a complexidade do cenário de classificação avaliado, possuindo 16 estados operacionais e alguns sendo extremamente parecidos, a capacidade de alcançar 100% de exatidão em apenas 25 *timesteps* demonstra um resultado satisfatório. A qualidade desse resultado é confirmada pela Tabela 6 que mostra o resultado de outros trabalhos na literatura que utilizaram modelos de RNP para classificar acidentes nucleares. Nenhum dos trabalhos apresentados foi capaz de alcançar esse nível de

desempenho da FCN, considerando o número de *timesteps* e a quantidade de estados operacionais classificados.

		Classe Predita															
		blackout	blacksem	loca	mefwisem	mefwiso	mfwbr	mfwbrsem	mfwisem	mfwiso	mstmisem	mstmiso	normal	sgtr	stmlibr	triprea	triptur
Classe Real	blackout	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	blacksem	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	loca	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	mefwisem	0%	0%	0%	90%	10%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	mefwiso	0%	0%	0%	14%	86%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	mfwbr	0%	0%	0%	0%	0%	93%	7%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	mfwbrsem	0%	0%	0%	0%	0%	6%	94%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	mfwisem	0%	0%	0%	0%	0%	0%	0%	80%	20%	0%	0%	0%	0%	0%	0%	0%
	mfwiso	0%	0%	0%	0%	0%	0%	0%	12%	88%	0%	0%	0%	0%	0%	0%	0%
	mstmisem	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%
	mstmiso	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%
	normal	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%
	sgtr	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%
	stmlibr	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%
	triprea	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%
	triptur	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%

Figura 27. Matriz de confusão da FCN para o conjunto de teste com 5 *timesteps* (De autoria própria)

		Classe Predita															
		blackout	blacksem	loca	mefwisem	mefwiso	mfwbr	mfwbrsem	mfwisem	mfwiso	mstmisem	mstmiso	normal	sgtr	stmlibr	triprea	triptur
Classe Real	blackout	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	
	blacksem	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	
	loca	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	
	mefwisem	0%	0%	0%	94%	6%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	
	mefwiso	0%	0%	0%	8%	92%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	
	mfwbr	0%	0%	0%	0%	0%	99%	1%	0%	0%	0%	0%	0%	0%	0%	0%	
	mfwbrsem	0%	0%	0%	0%	0%	3%	97%	0%	0%	0%	0%	0%	0%	0%	0%	
	mfwisem	0%	0%	0%	0%	0%	0%	0%	84%	16%	0%	0%	0%	0%	0%	0%	
	mfwiso	0%	0%	0%	0%	0%	0%	0%	8%	92%	0%	0%	0%	0%	0%	0%	
	mstmisem	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	
	mstmiso	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	
	normal	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	
	sgtr	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	
	stmlibr	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	
	triprea	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	
	triptur	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%

Figura 28. Matriz de confusão da FCN para o conjunto de teste com 10 *timesteps* (De autoria própria)

		Classe Predita															
		blackout	blacksem	loca	mefwisem	mefwiso	mfwbr	mfwbrsem	mfwisem	mfwiso	mstmisem	mstmiso	normal	sgtr	stmlibr	triprea	triptur
Classe Real	blackout	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	blacksem	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	loca	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	mefwisem	0%	0%	0%	95%	5%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	mefwiso	0%	0%	0%	7%	93%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	mfwbr	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	mfwbrsem	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	mfwisem	0%	0%	0%	0%	0%	0%	0%	94%	6%	0%	0%	0%	0%	0%	0%	0%
	mfwiso	0%	0%	0%	0%	0%	0%	0%	9%	91%	0%	0%	0%	0%	0%	0%	0%
	mstmisem	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%
	mstmiso	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%
	normal	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%
	sgtr	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%
	stmlibr	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%
	triprea	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%
triptur	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	

Figura 29. Matriz de confusão da FCN para o conjunto de teste com 15 *timesteps* (De autoria própria)

		Classe Predita														
		blackout	blacksem	loca	mefwisem	mefwiso	mfwbr	mfwbrsem	mfwisem	mfwiso	mstmisem	mstmiso	normal	sgtr	stmlibr	triprea
Classe Real	blackout	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	blacksem	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	loca	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	mefwisem	0%	0%	0%	99%	1%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	mefwiso	0%	0%	0%	6%	94%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	mfwbr	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	mfwbrsem	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%
	mfwisem	0%	0%	0%	0%	0%	0%	0%	99%	1%	0%	0%	0%	0%	0%	0%
	mfwiso	0%	0%	0%	0%	0%	0%	0%	4%	96%	0%	0%	0%	0%	0%	0%
	mstmisem	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%
	mstmiso	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%
	normal	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%
	sgtr	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%
	stmlibr	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%
	triprea	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%
triptur	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%

Figura 30. Matriz de confusão da FCN para o conjunto de teste com 20 *timesteps* (De autoria própria)

		Classe Predita														
		blackout	blacksem	loca	mefwisem	mefwiso	mfwbr	mfwbrsem	mfwisem	mfwiso	mstmisem	mstmiso	normal	sgtr	stmlibr	triprea
Classe Real	blackout	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	blacksem	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	loca	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	mefwisem	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	mefwiso	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	mfwbr	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%	0%
	mfwbrsem	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%
	mfwisem	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%
	mfwiso	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%
	mstmisem	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%
	mstmiso	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%
	normal	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%
	sgtr	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%
	stmlibr	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%	0%
	triprea	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%
triptur	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%

Figura 31. Matriz de confusão da FCN para o conjunto de teste com 25 *timesteps* (De autoria própria)

Tabela 6. Comparação do Resultado de Diferentes Trabalho na Tarefa de Classificação de Acidentes Nucleares

Trabalho	Modelo de Rede Neural Profunda	N ^a de Estados Operacionais	Timesteps	Exatidão
Peng <i>et al.</i> , (2018)	Deep Belief Network	7	-	99,10%
Santos <i>et al.</i> , (2019a)	Deep Rectifier Neural Network	13	-	99,03%
Santos <i>et al.</i> , (2019b)	One-Class Deep Autoecoders	13	-	96,18%
Yang <i>et al.</i> , (2020)	Long Short Term Memory	4	30	99,84%
Saeed <i>et al.</i> , (2020)	Convolutional Neural Network e Long Short-Term Memory	7	100	100%
Tese / Santos et al., (2021)	Fully Convolutional Neural Network	16	25	100%

4.3. EXPERIMENTOS PARA O MÓDULO DE VALIDAÇÃO

Nos experimentos para o módulo de validação, foram avaliadas duas abordagens para gerar resposta “Não Sei” utilizando LSTM-Autoencoder.

- **Abordagem All-Classes Long Short-Term Memory-Autoencoder (AC-LSTM-AE):** Consiste em treinar um modelo de AE-LSTM com todas as classes (acidentes) conhecidas.
- **Abordagem One-Class Long Short-Term Memory-Autoencoder (OC-LSTM-AE):** Consiste em treinar múltiplos modelos de AE-LSTM um para cada classe (acidente) conhecido.

Em ambas as abordagens, foi utilizada a técnica de janela deslizante para modelar os dados para os LSTM-AEs. Sendo assim, no primeiro teste comparativo entre as abordagens o conjunto de dados foi modelado com a técnica de janela deslizante com 5 *timesteps*. Ademais, como no total existem 16 estados operacionais, foram realizados 16 experimentos para cada abordagem.

Na abordagem AC-LSTM-AE, em cada experimento, um estado operacional foi retirado do conjunto de dados, os dados dos 15 estados operacionais restantes foram então utilizados para treinar um modelo de LSTM-AE. Após o treinamento, o erro de reconstrução máximo para o conjunto de treinamento foi utilizado para definir um valor de limiar. Em seguida, para avaliar a capacidade do modelo, este foi confrontado com exemplos do conjunto de teste das 15 classes conhecidas (treinadas) e com exemplos da classe desconhecida (não treinada), com a premissa que para os exemplos das classes

conhecidas o modelo deveria gerar um erro de reconstrução menor que o limiar, e para exemplos da classe desconhecida, deviria produzir um erro maior que o limiar.

Já na abordagem com OC-LSTM-AE, em cada experimento, um modelo de LSTM-AE foi treinado com dados de apenas um estado operacional do conjunto de dados. Após o treinamento, o erro de reconstrução máximo para o conjunto de treinamento também foi utilizado para definir um valor de limiar. Em seguida, para avaliar a capacidade do OC-LSTM-AE, este foi confrontado com exemplos do conjunto de teste da classe conhecida (treinada) e com exemplos das classes desconhecidas (não treinadas). A premissa, era mesma da abordagem anterior, para os exemplos da classe conhecida o modelo deveria gerar um erro de reconstrução menor que o limiar, e para exemplos das classes desconhecidas, deviria produzir um erro maior que o limiar.

4.3.1. Resultados

O resultado, mostrado na Figura 32, obtido com os experimentos utilizando a abordagem AC-LSTM-AE, revela que essa abordagem tem grande capacidade de identificar os eventos conhecidos, este fato é representado pela “Média Sei”, onde uma exatidão média de cerca de 99,998% foi alcançada na identificação dos eventos conhecidos. Em contrapartida, na identificação dos eventos desconhecidos, o desempenho desta abordagem foi extremamente limitado, alcançando exatidão média de apenas 54,314%; este valor é mostrado na “Média Não Sei”.

Já na Figura 33, é mostrado o resultado obtido com os experimentos usando a abordagem OC-LSTM-AE. Diferentemente da abordagem AC-LSTM-AE, que só alcançou um bom desempenho na identificação dos eventos conhecidos, a abordagem OC-LSTM-AE obteve um resultado satisfatório tanto na identificação dos eventos conhecidos (exatidão média de 99,76%), quanto na identificação dos eventos desconhecidos (exatidão média de 99,25%).

Dados os resultados superiores alcançados pela abordagem OC-LSTM-AE, experimentos adicionais foram realizados para avaliar se seria possível refinar ainda mais a sua capacidade de identificar eventos desconhecidos, sem afetar negativamente a sua capacidade de identificar os eventos conhecidos. Sendo assim, para esses experimentos, foram utilizados os conjuntos de dados modelados com a técnica de janela deslizante utilizando 10, 15, 20 e 25 *timesteps*.

As Figuras 34, 35, 36 e 37 mostram os resultados obtidos nos experimentos para cada valor de *timestep*. Assim como aconteceu com os modelos de RNP utilizados na classificação de acidentes nucleares, o desempenho da abordagem OC-LSTM-AE tendeu a melhorar de acordo com o aumento no valor de *timestep*, de modo que, com 25 *timesteps*, a abordagem OC-LSTM-AE, conseguiu alcançar uma exatidão média na identificação de eventos desconhecidos de 100%, e além disso, a capacidade de identificação dos eventos conhecidos não foi impactada negativamente, na verdade, a exatidão média na identificação dos eventos conhecidos alcançou um patamar de 99,85%.

Ao comparar os resultados obtidos utilizando a abordagem OC-LSTM-AE com outros similares na literatura (Tabela 7), pode se afirmar que a abordagem OC-LSTM-AE alcançou um desempenho do nível do estado da arte, nesse problema de gerar resposta “Não Sei” no contexto do PIDAN. Essa afirmação é ainda mais embasada ao considerar a complexidade do cenário apresentado, 16 estados operacionais, com alguns estados sendo extremamente parecidos.

	No blackout	No blacksem	No loca	No mefwisem	No mefwiso	No mfwbr	No mfwbrsem	No mfwisem	No mfwiso	No mstmisem	No mstmiso	No normal	No sgtr	No stmlibr	No triprea	No triptur
blackout	86,18%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
blacksem	100%	100,00%	100%	100%	100%	99,98%	99,96%	99,96%	100%	100%	100%	100%	100%	100%	100%	100%
loca	100%	100%	67,69%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
mefwisem	100%	100%	100%	15,11%	100%	100%	100%	99,98%	100%	100%	100%	100%	100%	100%	100%	100%
mefwiso	100%	100%	100%	100%	10,64%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
mfwbr	100%	100%	100%	100%	100%	86,89%	100%	99,98%	100%	100%	100%	100%	100%	100%	100%	100%
mfwbrsem	100%	100%	100%	100%	100%	100%	82,33%	99,98%	100%	100%	100%	100%	100%	100%	100%	100%
mfwisem	100%	100%	100%	100%	100%	100%	100%	67,27%	100%	100%	100%	100%	100%	100%	100%	100%
mfwiso	100%	100%	100%	100%	100%	100%	100%	100%	24,55%	100%	99,96%	100%	100%	100%	100%	100%
mstmisem	100%	100%	100%	100%	100%	100%	100%	99,96%	100%	88,25%	100,00%	100%	100%	100%	100%	100%
mstmiso	100%	100%	100%	100%	100%	100%	100%	99,98%	100%	100%	44,55%	100%	100%	100%	100%	100%
normal	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	0,00%	100%	100%	100%	100%
sgtr	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	69,45%	100%	100%	100%
stmlibr	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	0,00%	100%	100%
triprea	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	47,56%	100%
triptur	100%	100%	100%	100%	100%	100%	99,98%	100%	100%	100%	100%	100%	99,98%	100%	100%	78,55%
Média Não Sei	54,314%															
Média Sei	99,998%															

Figura 32. Resultados dos experimentos com a abordagem AC-LSTM-AE utilizando um conjunto de dados com 5 *timesteps* (De autoria própria)

	AE blackout	AE blacksem	AE loca	AE mefwisem	AE mefwiso	AE mfwbr	AE mfwbrsem	AE mfwisem	AE mfwiso	AE mstmisem	AE mstmiso	AE normal	AE sgtr	AE stmlibr	AE triprea	AE triptur
blackout	99,98%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
blacksem	100%	99,93%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
loca	100%	100%	100,00%	100%	100%	100%	100%	100%	100%	100%	100%	100%	88,51%	100%	100%	100%
mefwisem	100%	100%	100%	99,84%	67,24%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
mefwiso	100%	100%	100%	67,31%	99,56%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
mfwbr	100%	100%	100%	100%	100%	99,49%	87,33%	100%	100%	100%	100%	100%	100%	100%	100%	100%
mfwbrsem	100%	100%	100%	100%	100%	87,47%	99,44%	100%	100%	100%	100%	100%	100%	100%	100%	100%
mfwisem	100%	100%	100%	99,07%	99,96%	100%	100%	99,44%	68,09%	100%	100%	100%	100%	100%	100%	100%
mfwiso	100%	100%	100%	99,40%	99,98%	100%	100%	69,13%	99,96%	100%	100%	100%	100%	100%	100%	100%
mstmisem	100%	100%	100%	100%	100%	100%	100%	100%	100%	99,38%	100%	100%	100%	100%	100%	100%
mstmiso	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	99,78%	100%	100%	100%	94,35%	100%
normal	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	99,98%	100%	100%	100%	100%
sgtr	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100,00%	100,00%	100%	100%
stmlibr	100%	100%	99,52%	100%	100%	100%	100%	100%	100%	100%	100%	100%	96,71%	99,84%	100%	100%
triprea	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	96,76%	100%	100%	100%	99,85%	100%
triptur	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	99,76%
Média Não Sei	99,25%															
Média Sei	99,76%															

Figura 33. Resultados dos experimentos com a abordagem OC-LSTM-AE utilizando um conjunto de dados com 5 *timesteps* (De autoria própria)

	AE blackout	AE blacksem	AE loca	AE mefwisem	AE mefwiso	AE mfwbr	AE mfwbrsem	AE mfwisem	AE mfwiso	AE mstmisem	AE mstmiso	AE normal	AE sgtr	AE stmlibr	AE triprea	AE triptur
blackout	99,94%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
blacksem	100%	99,90%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
loca	100%	100%	100,00%	100%	100%	100%	100%	100%	100%	100%	100%	100%	98,16%	100%	100%	100%
mefwisem	100%	100%	100%	99,98%	71,86%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
mefwiso	100%	100%	100%	74,00%	99,92%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
mfwbr	100%	100%	100%	100%	100%	99,49%	96,00%	100%	100%	100%	100%	100%	100%	100%	100%	100%
mfwbrsem	100%	100%	100%	100%	100%	95,10%	99,78%	100%	100%	100%	100%	100%	100%	100%	100%	100%
mfwisem	100%	100%	100%	78,96%	82,58%	100%	100%	99,44%	73,30%	100%	100%	100%	100%	100%	100%	100%
mfwiso	100%	100%	100%	85,02%	81,24%	100%	100%	76,00%	99,96%	100%	100%	100%	100%	100%	100%	100%
mstmisem	100%	100%	100%	100%	100%	100%	100%	100%	100%	99,38%	100%	100%	100%	100%	100%	100%
mstmiso	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	99,94%	100%	100%	100%	77,04%	100%
normal	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100,00%	100%	100%	100%	100%
sgtr	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100,00%	100,00%	100%	100%
stmlibr	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	99,94%	100,00%	100%	100%
triprea	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	83,76%	100%	100%	100%	99,94%	100%
triptur	100%	100%	100%	100%	100%	100%	100%	100%	100%	93%	100%	100%	100%	100%	100%	99,90%
Média Não Sei	99,02%															
Média Sei	99,85%															

Figura 34. Resultados dos experimentos com a abordagem OC-LSTM-AE utilizando um conjunto de dados com 10 *timesteps* (De autoria própria)

	AE blackout	AE blacksem	AE loca	AE mefwisem	AE mefwiso	AE mfwbr	AE mfwbrsem	AE mfwisem	AE mfwiso	AE mstmisem	AE mstmiso	AE normal	AE sgtr	AE stmlibr	AE triprea	AE triptur
blackout	99,76%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
blacksem	100%	99,93%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
loca	100%	100%	100,00%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	98,16%	100%	100%
mefwisem	100%	100%	100%	100,00%	80,18%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
mefwiso	100%	100%	100%	82,22%	99,89%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
mfwbr	100%	100%	100%	100%	100%	99,91%	96,00%	100%	100%	100%	100%	100%	100%	100%	100%	100%
mfwbrsem	100%	100%	100%	100%	100%	100,00%	99,44%	100%	100%	100%	100%	100%	100%	100%	100%	100%
mfwisem	100%	100%	100%	87,22%	96,67%	100%	100%	99,96%	82,00%	100%	100%	100%	100%	100%	100%	100%
mfwiso	100%	100%	100%	90,67%	95,47%	100%	100%	82,87%	99,93%	100%	100%	100%	100%	100%	100%	100%
mstmisem	100%	100%	100%	100%	100%	100%	100%	100%	100%	99,98%	100%	100%	100%	100%	100%	100%
mstmiso	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	99,76%	100%	100%	100%	100%	100%
normal	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	99,98%	100%	100%	100%	100%
sgtr	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	99,96%	100,00%	100%	100%
stmlibr	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	99,94%	99,87%	100%	100%
triprea	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	99,82%	100%	100%	99,94%	100%
triptur	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100,00%
Média Não Sei	99,55%															
Média Sei	99,89%															

Figura 35. Resultados dos experimentos com a abordagem OC-LSTM-AE utilizando um conjunto de dados com 15 *timesteps* (De autoria própria)

	AE blackout	AE blacksem	AE loca	AE mefwisem	AE mefwiso	AE mfwbr	AE mfwbrsem	AE mfwisem	AE mfwiso	AE mstmisem	AE mstmiso	AE normal	AE sgtr	AE stmlibr	AE triprea	AE triptur
blackout	99,43%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
blacksem	100%	99,95%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
loca	100%	100%	100,00%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
mefwisem	100%	100%	100%	99,70%	89,85%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
mefwiso	100%	100%	100%	92,50%	100,00%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
mfwbr	100%	100%	100%	100%	100%	99,83%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
mfwbrsem	100%	100%	100%	100%	100%	100%	99,68%	100%	100%	100%	100%	100%	100%	100%	100%	100%
mfwisem	100%	100%	100%	99,00%	98,02%	100%	100%	99,95%	92,38%	100%	100%	100%	100%	100%	100%	100%
mfwiso	100%	100%	100%	99,35%	94,80%	100%	100%	92,50%	99,80%	100%	100%	100%	100%	100%	100%	100%
mstmisem	100%	100%	100%	100%	100%	100%	100%	100%	100%	99,75%	100%	100%	100%	100%	100%	100%
mstmiso	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	99,92%	100%	100%	100%	100%	100%
normal	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	99,92%	100%	100%	100%	100%
sgtr	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	99,97%	100,00%	100%	100%
stmlibr	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	99,95%	100%	100%
triprea	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100,00%	100%
triptur	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	99,80%
Média Não Sei	99,83%															
Média Sei	99,85%															

Figura 36. Resultados dos experimentos com a abordagem OC-LSTM-AE utilizando um conjunto de dados com 20 *timesteps* (De autoria própria)

	AE blackout	AE blacksem	AE loca	AE mefwisem	AE mefwiso	AE mfwbr	AE mfwbrsem	AE mfwisem	AE mfwiso	AE mstmisem	AE mstmiso	AE normal	AE sgtr	AE stmlibr	AE triprea	AE triptur
blackout	99,77%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
blacksem	100%	100,00%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
loca	100%	100%	99,94%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
mefwisem	100%	100%	100%	98,91%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
mefwiso	100%	100%	100%	100,00%	99,63%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
mfwbr	100%	100%	100%	100%	100%	99,83%	100,00%	100%	100%	100%	100%	100%	100%	100%	100%	100%
mfwbrsem	100%	100%	100%	100%	100%	100,00%	99,69%	100%	100%	100%	100%	100%	100%	100%	100%	100%
mfwisem	100%	100%	100%	100%	100%	100%	100%	100,00%	100%	100%	100%	100%	100%	100%	100%	100%
mfwiso	100%	100%	100%	100%	100%	100%	100%	100%	100,00%	100%	100%	100%	100%	100%	100%	100%
mstmisem	100%	100%	100%	100%	100%	100%	100%	100%	100%	99,97%	100%	100%	100%	100%	100%	100%
mstmiso	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	99,94%	100%	100%	100%	100%	100%
normal	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	99,91%	100%	100%	100%	100%
sgtr	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	99,94%	100,00%	100%	100%
stmlibr	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	99,86%	100%	100%
triprea	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100,00%	100%
triptur	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	99,77%
Média Não Sei	100,00%															
Média Sei	99,82%															

Figura 37. Resultados dos experimentos com a abordagem OC-LSTM-AE utilizando um conjunto de dados com 25 *timesteps* (De autoria própria)

Tabela 7. Comparação do Resultado de Diferentes Trabalho na Tarefa de Gerar Resposta “Não Sei”

Trabalho	Modelo de Rede Neural Profunda	Número de Estados Operacionais	<i>Timesteps</i>	Exatidão Média no Conjunto Conhecido	Exatidão Média no Conjunto Desconhecido
Pinheiro <i>et al.</i> , (2020)	One-Class Deep Autoencoders	13	-	94,56%	99,88%
Yang <i>et al.</i> , (2020)	AC-LSTM-AE	4	-	-	-
Tese / Santos <i>et al.</i> , (2021)	OC-LSTM-AE	16	5	99,76%	99,25%
			10	99,85%	99,02%
			15	99,89%	99,55%
			20	99,85%	99,83%
			25	99,82%	100%

*No trabalho de Yang *et al.*, (2020) não é fornecida uma métrica que demonstre o nível de exatidão de sua abordagem.

CAPÍTULO 5

CONCLUSÕES

Embora o desenvolvimento de sistemas baseados em técnicas de inteligência artificial para identificação e diagnóstico de acidentes em usinas nucleares seja um tema que, através das décadas, foi vastamente estudado, o advento do aprendizado profundo, o paradigma moderno das redes neurais artificiais, está permitindo que tais sistemas alcancem níveis de desempenho nunca alcançados previamente. Os modelos contemporâneos de rede neural profunda, como mostram (SANTOS *et al.*, 2019a; SANTOS *et al.*, 2019b; PINHEIRO *et al.*, 2020; SANTOS *et al.*, 2021), quando utilizados para realizar as tarefas principais do problema de identificação e diagnóstico de acidentes nucleares (PIDAN) (classificação de acidentes nucleares e geração de resposta “Não Sei”), conseguem alcançar um desempenho do nível do estado da arte.

Contudo, apesar dos resultados promissores que os modelos profundos conseguem alcançar nas tarefas principais de um sistema para o PIDAN, um dos problemas fundamentais desse tipo de sistema permanece, sendo a estruturação do sistema de forma que este alcance um exímio nível de exatidão (próximo dos 99%), tanto quando confrontado com dados das classes conhecidas, quanto quando confrontado com dados desconhecidos (nesse caso gerando resposta “Não Sei”).

Sendo assim, a inovação desta tese é o desenvolvimento de um sistema com estrutura modular, onde cada módulo executa uma tarefa específica do PIDAN por meio de um ou mais modelos de aprendizagem profunda. Esta estruturação do sistema, utilizando módulos individuais, além de garantir um alto nível de exatidão via os modelos de RNP apresentados, também evita que, por exemplo, um alto nível de exatidão na identificação de dados de classes conhecidas, influencie negativamente a detecção de dados desconhecidos.

Em função da extensa pesquisa bibliográfica realizada, este é o primeiro estudo que propõe o desenvolvimento de um sistema modular cujos módulos implantam um ou mais modelos contemporâneos de RNP como base principal na resolução das suas respectivas tarefas. Além disso, este é o primeiro estudo que se propõe a avaliar de forma aprofundada e comparativa o desempenho de diferentes modelos e abordagens baseadas em RNPs, considerados estado da arte, na resolução das tarefas principais do PIDAN.

Os resultados obtidos demonstram o potencial e relevância da pesquisa realizada nesta tese, pois dada a complexidade do estudo de caso utilizado para os experimentos com um sistema modular proposto, é possível afirmar que as abordagens propostas neste trabalho, tanto para classificação de acidentes nucleares, quanto para a geração de resposta “Não Sei” conseguiram alcançar um desempenho do nível do estado da arte. Ademais, os resultados obtidos mostram que o sistema proposto apresenta um alto nível de confiabilidade, eficiência e velocidade de processamento. Sendo assim, o sistema e os estudos realizados nesta tese representam um passo importante no desenvolvimento de um sistema com capacidade de resolver o PIDAN, que possa ser realmente implantado para auxiliar os operadores na tarefa de tomada de decisão, na ocorrência de um evento anômalo durante a operação de uma usina nuclear.

Ademais, o sistema desenvolvido possui relevância para além da área nuclear, uma vez que este tipo de sistema, ao ser baseado em modelos de aprendizado profundo, pode ser adaptado para ser aplicado em diversas outras áreas, como a medicina, a indústria automotiva, a aviação, entre outras. Em todos esses campos, a tomada de decisão é um aspecto fundamental para a segurança e eficiência do processo, e um sistema capaz de identificar e diagnosticar problemas de forma precisa e confiável pode ajudar a evitar acidentes e prejuízos. Além disso, a capacidade de responder "não sei" quando confrontado com dados desconhecidos, é uma funcionalidade de grande importância para evitar erros de diagnóstico e tomada de decisão baseados em informações incorretas ou incompletas. Portanto, um sistema de identificação e diagnóstico, baseado em aprendizagem profunda, e com capacidade de gerar resposta “Não Sei” pode trazer benefícios significativos para diversas áreas de aplicação.

Neste sentido, para trabalhos futuros, sugere-se aprimorar esses sistemas adicionando a capacidade de identificar as variáveis de estado mais relevantes para determinado estado operacional, bem como a capacidade de prever e alertar sobre possíveis estados operacionais antes que eles ocorram. Essas melhorias tornariam o sistema ainda mais valioso para a segurança e eficiência das usinas nucleares.

REFERÊNCIAS BIBLIOGRÁFICAS

ABADI, M. et al. TensorFlow: A system for large-scale machine learning. OSDI'16: Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation. p. 265-283, 2016.

ALBAWI, S. Q. F Social Touch Gesture Recognition Using Deep Neural Network. 2018. 110 f. (Tese – Doutorado em Engenharia da Computação e Elétrica) Altinbaş University.

ALESHIN-GUENDEL, S. Examining the Structure of Convolutional Neural Networks. 2017. 1–30 f. Boston College, Chestnut Hill, 2017.

ALVARENGA, M. A. B. Diagnóstico do desligamento de um reator nuclear através de técnicas avançadas de inteligência artificial. 1997. Universidade Federal do Rio de Janeiro, 1997.

ALVES, A. C. P. D. Um Sistema de Análise de “Trip” em Reatores PWR Usando Redes Neurais. 1993. 147 f. (Tese – Doutorado em Engenharia Nuclear) Universidade Federal do Rio de Janeiro.

BAGNALL, A. et al. The UEA multivariate time series classification archive. arXiv:1811.00075, 2018.

BAHDANAU, D.; CHO, K. H.; BENGIO, Y. Neural machine translation by jointly learning to align and translate. arXiv: 1409.0473, 2015.

BARTLETT, E. B.; UHRIG, R. E. Nuclear power plant status diagnostics using an artificial neural network. Nuclear Technology, v. 97, n. 3, p. 272–281, 1992.

BATISTA, R. A. Busca de Padrões em Séries Temporais. 2010. Universidade Federal do Rio de Janeiro, 2010.

BEJNORDI, B. E. et al. Diagnostic assessment of deep learning algorithms for

detection of lymph node metastases in women with breast cancer. *JAMA - Journal of the American Medical Association*, v. 318, n. 22, p. 2199–2210, 2017.

BENGIO, Y.; SIMARD, P.; FRASCONI, P. Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, v. 5, n. 2, p. 157-166, 1994.

BENUWA, B. B. et al. A review of deep machine learning. *International Journal of Engineering Research in Africa*, v. 24, p. 124-136, 2016.

BRACEWELL, R. *The Fourier Transform & Its Applications*. third edit ed.: McGraw-Hill Science/Engineering/Math; 3 edition, 1999.

BROUGHTON, J. M. et al. Scenario of the Three Mile Island Unit 2 accident. *Nuclear Technology*, v. 87, n. 1, p. 34–53, 1989.

BROWN, T. B. et al. Language Models are Few-Shot Learners. *arXiv:2005.14165*, 2020.

CHALAPATHY, R.; CHAWLA, S. Deep Learning for Anomaly Detection: A Survey. *arXiv: 1901.03407*, 2019.

CHALAPATHY, R.; MENON, A. K.; CHAWLA, S. Anomaly Detection using One-Class Neural Networks. *arXiv:1802.06360*, 2018.

CHENG, Y. H. et al. Introducing PCTTRAN as an evaluation tool for nuclear power plant emergency responses. *Annals of Nuclear Energy*, v. 40, n. 1, p. 122–129, 2012.

COUDRAY, N. et al. Classification and mutation prediction from non–small cell lung cancer histopathology images using deep learning. *Nature Medicine*, v. 24, n. 10, p. 1559–1567, 2018.

CUI, Z. et al. Multi-scale convolutional neural networks for time series classification. *ArXiv 1603.06995*, 2016.

DAM, R.S.F., et al. A novel radioactive particle tracking algorithm based on deep rectifier neural network. *Nucl. Eng. Technol.*, 1738–5733, 2021.

DAU, H. A. et al. The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica*, v. 6, n. 6, p. 1293–1305, 2019.

DENG, L. et al. Recent advances in deep learning for speech research at Microsoft. *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2013)*, 2013.

DESTERRO, F.S.M. et al. Development of a Deep Rectifier Neural Network for dose prediction in nuclear emergencies with radioactive material releases. *Progress in Nuclear Energy*. v. 118, p. 103110, 2020.

DU, M. et al. DeepLog: Anomaly detection and diagnosis from system logs through deep learning. *CCS '17: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, p. 1285–1298, 2017.

ESTEVA, A. et al. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, v. 542, n. 7639, p. 115–118, 2017.

FACURE, M. Funções de Ativação. 2017. Disponível em: <<https://matheusfacure.github.io/2017/07/12/activ-func/>>. Acesso em: 03 abr. 2023.

FAWAZ, I. H. et al. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, v. 33, n. 4, p. 917–963, 2019.

FUKUSHIMA, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, v. 36, n. 4, p. 193–202, 1980.

GÉRON, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2nd Edition [Book]. 2nd Edition ed.: O'Reilly Media, Inc., 2019.

GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010), v. 9, p. 249–256, 2010.

GLOROT, X.; BORDES, A.; BENGIO, Y. Deep sparse rectifier neural networks. In Proc. 14th International Conference on Artificial Intelligence and Statistics, p. 315–323, 2011.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep Learning. MIT Press, 2016.

GUTOSKI, M. et al. A clustering-based deep autoencoder for one-class image classification. IEEE Latin American Conference on Computational Intelligence (LACCI 2017), Arequipa, p. 1–6, 2017.

HAHNLOSER, R. H. R. et al. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. Nature, 405, p. 947-951, 2000.

HANNUN, A. et al. Deep Speech: Scaling up end-to-end speech recognition. arXiv:1412.5567, 2014.

HEBB, D. O. The Organization of Behavior. Wiley, New York, 1949.

HERZOG, M. H.; CLARKE, A. M. Why vision is not both hierarchical and feedforward. Frontiers in Computational Neuroscience, v. 8, 2014.

HINTON, G. E.; GEOFFREY E.; OSINDERO, S.; TEH, Y. W. A fast learning algorithm for deep belief nets. Neural Computation, 18:1527–1554, 2006.

HINTON, G. E.; MCCLELLAND J; RUMELHART D. Distributed representations. Parallel Distributed Processing: Explorations in the Microstructure of Cognition. volume 1 ed. Cambridge: MIT Press, p. 77–109, 1986.

HINTON, G. E.; SALAKHUTDINOV, R. R. Reducing the dimensionality of data with

neural networks. *Science*, 313 (5786):504–507, 2006.

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-Term Memory. *Neural Computation*, 9(8),1735–1780,1997.

HOCHREITER S. Untersuchungen zu dynamischen neuronalen Netzen. 1991. Technische Universität München, 1991.

HU, J.; SHEN, L.; SUN, G. Squeeze-and-Excitation Networks. arXiv:1709.01507, 2018.

HUBEL, D. H.; WIESEL, T. N. Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology*, v. 195, n. 1, p. 215–243, 1968.

HUBEL, D. H.; WIESEL, T. N. Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, v. 148, n. 3, p. 574–591, 1959.

JEONG, E.; FURUTA, K.; KONDO, S. Identification of Transient in Nuclear Power Plant Using Adaptive Template Matching with Neural Network. Proc. International Topical Meeting on Nuclear Power Plant, Instrumentation, Control and Human-Machine Interface Technologies, State College, Pennsylvania, May 6-9, 1996, Vol. 1, p. 243, American Nuclear Society, La Grange Park, Illinois, 1996.

JOYCE, J. P.; LAPINSKY, G. W. A history and overview of the safety parameter display system concept. *IEEE Transactions on Nuclear Science*, v. 30, n. 1, p. 744–749, 1983.

KARIM, F. et al. LSTM Fully Convolutional Networks for Time Series Classification. *IEEE access*, 6:1662–1669, 2017.

KARIM, F. et al. Multivariate LSTM-FCNs for time series classification. *Neural Networks*, 116:237245, 2019.

KRAMER, M. A. Autoassociative neural networks. *Computers and Chemical*

Engineering, v. 16, p. 313-328, 1992.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. AlexNet. NIPS, p. 1106-11144, 2012.

KWON, D. et al. A survey of deep learning-based network anomaly detection. Cluster Computing, v. 22, p. 949-961, 2019.

KWON, K. C.; KIM, J. H. Accident identification in nuclear power plants using hidden Markov models. Engineering Applications of Artificial Intelligence, v. 12, n. 4, p. 491–501, 1999.

LECUN, Y. Generalization and network design strategies. Technical Report CRG-TR-89-4, University of Toronto, 1989.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. Nature, 521(7553), pp 436-444, 2015.

LEE, D.; SEONG, P. H.; KIM, J. Autonomous operation algorithm for safety systems of nuclear power plants by using long-short term memory and function-based hierarchical framework. Annals of Nuclear Energy, v. 119, p. 287-299, 2018.

LE GUENNEC, A. et al. Data augmentation for time series classification using convolutional neural networks. In: ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data, 2016.

LEWIS, H. W. et al. Risk assessment review group report to the US Nuclear Regulatory Commission IEEE Trans Nucl Sci, 26 (5), p. 4686-4690, 1979.

LONG, J.; SHELHAMER, E.; DARRELL, T. Fully convolutional networks for semantic segmentation. arXiv:1411.4038, 2015.

MARKOU, M.; SINGH, S. Novelty detection: A review - Part 1: Statistical approaches. Signal Processing, 83(12), p. 2481-2497, 2003.

MINSKY, M.; PAPERT, S. A. Perceptrons An Introduction to Computational Geometry, 1969.

MAYR, A. et al. DeepTox: Toxicity prediction using deep learning. *Frontiers in Environmental Science*, v. 3, 2016.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 1943.

MOEN, E. et al. Deep learning for cellular image analysis. *Nature Methods*, v. 16, p. 1233-1246, 2019.

MOL, A. C. D. A.; MARTINEZ, A. S.; SCHIRRU, R. A neural model for transient identification in dynamic processes with “don’t know” response. *Annals of Nuclear Energy*, v. 30, n. 13, p. 1365–1381, 2003.

MUHAMAD, Y. et al. Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry’s Nail. *Journal of Physics: Conference Series*, 2019.

NICOLAU, A. S.; SCHIRRU, R. A new methodology for diagnosis system with ‘Don’t Know’ response for Nuclear Power Plant. *Annals of Nuclear Energy*, v. 100, p. 91–97, 2017.

OLAH, C. Understanding LSTM Networks. 2015. Disponível em: <<https://colah.github.io/posts/2015-08-Understanding-LSTMs/>>. Acesso em: 03 abr. 2023.

OZA, P.; PATEL, V. M. Active authentication using an autoencoder regularized CNN-based one-class classifier. arXiv:1903.01031, 2019.

PARK, J. et al. Deep Learning Inference in Facebook Data Centers: Characterization, Performance Optimizations and Hardware Implications. arXiv:1811.09886, 2018.

PEDAMONTI, D. Comparison of Non-linear Activation Functions for Deep Neural Networks on MNIST Classification Task. arXiv:1804.02763, 2018.

PENG, B-S. et al. Research on intelligent fault diagnosis method for nuclear power plant based on correlation analysis and deep belief network. Progress in Nuclear Energy, v. 108, p. 419-427, 2018.

PEREIRA, C. M. N. A.; SCHIRRU, R.; MARTINEZ, A. S. Learning an optimized classification system from a data base of time series patterns using genetic algorithms. Transactions on Information and Communications Technologies, v. 22, p. 14, 1998.

PERERA, P. et al. OCGAN: One-class Novelty Detection Using GANs with Constrained Latent Representations. In: Conference on Computer Vision and Pattern Recognition (2019), 2019.

PEREZ, L.; WANG, J. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. arXiv: 1712.0462, 2017.

PEREZ, M. et al. The Development of RELAP/SCDAPSIM/MOD4.0 for Advanced Fluid Systems Design Analysis. 23th International Conference on Nuclear Engineering (ICONE-23), Chiba, Japan, May 17–21, p. 3737, 2015.

PIMENTEL, M. A. F. et al. A review of novelty detection. Signal Process, v. 99, p. 215–249, 2014.

PINHEIRO, V. H. C. 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, p. 107111, 2020.

PINHEIRO, V. H. C.; SCHIRRU, R. Genetic programming applied to the identification of accidents of a PWR nuclear power plant. Annals of Nuclear Energy, 2019.

POWLES, J.; HODSON, H. Google DeepMind and healthcare in an age of algorithms.

Health and Technology, v. 7, n. 4, p. 351–367, 2017.

QIU, X. et al. Ensemble deep learning for regression and time series forecasting. Proc. IEEE Symp. Comput. Intell. Ensemble Learn. (CIEL), p. 1-6, 2014.

RASCHKA, S.; MIRJALILI, V. Python Machine Learning - Second Edition: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow. Second Edition ed.: Packt Publishing, 2017.

ROCCO S., C. M.; ZIO, E. A support vector machine integrated system for the classification of operation anomalies in nuclear components and systems. Reliability Engineering and System Safety, v. 92, n. 5, p. 593–600, 2007.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65(6), 386–408, 1958.

RUFF, L. et al. Deep One-Class Classification. In: Proceedings of the 35th International Conference on Machine Learning, PMLR 80:4393-4402, 2018.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. Nature, v. 323, n. 6088, p. 533–536, 1986.

RUSSAKOVSKY, O. et al. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision, 2015.

SAEED, H. A. et al. Novel fault diagnosis scheme utilizing deep learning networks. Progress in Nuclear Energy, v. 118, p. 103066, 2020.

SAGHEER, A.; KOTB, M. Unsupervised Pre-training of a Deep LSTM-based Stacked Autoencoder for Multivariate Time Series Forecasting Problems. Scientific Reports, v. 9, n. 1, p. 1–16, 2019.

SANTHANAM, P.; FARCHI, E.; PANKRATIUS, V. Engineering Reliable Deep Learning Systems. arXiv:1910.12582, 2019.

SANTOS, M. C. et al. (2019a). Deep rectifier neural network applied to the accident identification problem in a PWR nuclear power plant. *Annals of Nuclear Energy*, v. 133, p. 400–408, 2019.

SANTOS, M. C. et al. (2019b). Identification of Postulated Accidents of a PWR Nuclear Power Plant using Deep Autoencoders. In: *International Nuclear Atlantic Conference - INAC*, 2019.

SANTOS, M. C. et al. (2021). A multiple-architecture deep learning approach for nuclear power plants accidents classification including anomaly detection and “don’t know” response. *Annals of Nuclear Energy*, v. 162, p. 108521, 2021.

SALGADO, C. M. et al. Development of a deep rectifier neural network for fluid volume fraction prediction in multiphase flows by gamma-ray densitometry. *Radiation Physics and Chemistry*, v. 189, p. 109708, 2021.

SAYEED, M. A. et al. Detecting Malaria from Segmented Cell Images of Thin Blood Smear Dataset using Keras from Tensorflow. *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, v. 8 (I), 2020.

SCHMIDHUBER, J. Deep Learning in neural networks: An overview. *Neural Networks*, v. 61, p.85-117, 2015.

SERRÁ, J. et al. Towards a universal neural network encoder for time series. *Artificial Intelligence Research and Development: Current Challenges, New Trends and Applications*, 308:120, 2018.

SHARKEY, N. E. J.A. Anderson and E. Rosenfeld (Eds.), *Talking Nets: An Oral History of Neural Networks*. *Artificial Intelligence*, v. 119, n. 1–2, p. 287–293, 2000.

SRIVASTAVA, R. K.; GREFF, K.; SCHMIDHUBER, J. Training Very Deep Networks. *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, p. 1–9, 2015.

STANFORD. Convolutional Neural Networks (CNNs / ConvNets). s.d. Disponível em: <<https://cs231n.github.io/convolutional-networks/>>. Acesso em: 03 abr. 2023.

SUSTO, G. A.; CENEDESE, A.; TERZI, M. Time-Series Classification Methods: Review and Applications to Power Systems Data. *Big Data Application in Power Systems*, p. 179–220, 2018.

SUTSKEVER, I.; VINYALS, O.; LE, Q. V. Sequence to Sequence Learning with Neural Networks. *Advances in Neural Information Processing Systems*, v. 4, p. 3104–3112, 2014.

TANISARO, P.; HEIDEMANN, G. Time series classification using time warping invariant echo state networks. In: *IEEE International Conference on Machine Learning and Applications*, pp 831–836, 2016.

TEIXEIRA, T.P. et al. Determination of eccentric deposition thickness on offshore horizontal pipes by gamma-ray densitometry and artificial intelligence technique. v. 165, p. 109221, 2020.

VASILEV, I.; SLATER, D.; SPACAGNA, G. *Python Deep Learning -Second Edition*. 2nd Revised edition ed. 2019.

VÁZQUEZ, F. *Deep Learning made easy with Deep Cognition*. 2017. Disponível em: <<https://becominghuman.ai/deep-learningmade-easy-with-deep-cognition-403fbe445351>>. Acesso em: 03 abr. 2023.

WANG, D. et al. *Deep Learning for Identifying Metastatic Breast Cancer*. arXiv:1606.05718, 2016.

WANG, H.; RAJ, B. *On the Origin of Deep Learning*. arXiv:1702.07800, 2017.

WANG, Z.; YAN, W.; OATES, T. Time series classification from scratch with deep neural networks: A strong baseline. *Institute of Electrical and Electronics Engineers Inc*,

p. 1578–1585, 2017.

WEI, Q. et al. Anomaly detection for medical images based on a one-class classification. SPIE, p. 57, 27, 2018.

WEN, Q. et al. Time Series Data Augmentation for Deep Learning: A Survey. arXiv:2002.12478, 2020.

WERBOS, P. J. Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. PhD Thesis, Harvard U., 1974.

WU, Y. et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. arXiv:1609.08144, 2016.

YAMASHITA, R. et al. Convolutional neural networks: an overview and application in radiology. Insights into Imaging, v. 9, p. 611-629, 2018.

YANG, J. et al. Nuclear Power Plant Accident Diagnosis Algorithm Including Novelty Detection Function Using LSTM. In: Ahram T. (eds) Advances in Artificial Intelligence, Software and Systems Engineering. AHFE 2019. Advances in Intelligent Systems and Computing, v. 965, 2020.

YANPING, C. et al. UCR Time Series Classification Archive, 2015. URL:www.cs.ucr.edu/~eamonn/time_series_data/.

ZHAO, B. et al. Convolutional neural networks for time series classification. Systems Engineering and Electronics 28(1):162–169, 2017.

ZHENG, Y. et al. Time series classification using multi-channels deep convolutional neural networks. In: Web-Age Information Management, pp 298–310, 2014.

ZHENG, Y. et al. Exploiting multi-channels deep convolutional neural networks for multivariate time series classification. Frontiers of Computer Science 10(1):96–112, 2016.

ZHOU, B. et al. Learning Deep Features for Discriminative Localization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, p. 2921–2929, 2016.

ZHOU, C.; PAFFENROTH, R. C. Anomaly Detection with Robust Deep Autoencoders. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 13-17, Halifax, NS, Canada. 2017.

ZHOU, Y. et al. Is Joint Training Better for Deep Auto-Encoders?. arXiv:1405.1380, 2014.