Universidade Federal
do Rio de Janeiro
Escola Politécnica

CLASSIFICAÇÃO DE ACIDENTES DE UMA USINA NUCLEAR DO TIPO PWR USANDO REDES NEURAIS

Lucas da Silva Moreira

Projeto de Graduação apresentado ao Curso de Engenharia nuclear da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientadora: Profª. Andressa dos Santos Nicolau

Rio de Janeiro
Janeiro de 2019

CLASSIFICAÇÃO DE ACIDENTES DE UMA USINA NUCLEAR DO TIPO PWR USANDO REDES NEURAIS

Lucas da Silva Moreira

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA NUCLEAR DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO NUCLEAR

Examinada por:

_____

Profª. Andressa dos Santos Nicolau, D.Sc

_____

Profº. Paulo Fernando Ferreira Frutuoso e Melo, D.Sc

_____

Profº. Roberto Schirru, D.Sc

RIO DE JANEIRO, RJ - BRASIL

JANEIRO de 2019

Moreira, Lucas da Silva

/ Lucas da Silva Moreira – Rio de Janeiro: UFRJ / Escola Politécnica, 2019.

XIII, 58 p.:il.; 29,7 cm.

Orientadora: Andressa dos Santos Nicolau.

Projeto de Graduação – UFRJ / Escola Politécnica / Curso de Engenharia nuclear, 2019.

Referências Bibliográficas: p. 55-58.

1. Redes Neurais. 2. Aprendizado de Máquina. 3. Acidentes em Centrais Nucleares. 4. Classificação de Acidentes.

Nicolau, Andressa dos Santos. II. Universidade Federal do Rio de Janeiro, Escola Politécnica, Curso Engenharia Nuclear. III. Classificação de Acidentes de uma Usina Nuclear do tipo PWR usando Redes Neurais

.

“Ainda que eu ande pelo vale da sombra da morte, não temerei mal algum, pois Tu estás comigo;...”

Salmo 23:4

DEDICATÓRIA

Dedico este trabalho à memória do grande físico, grande paraquedista, grande professor, grande amigo, grande ser humano, o grande Filipe Robert Sugaya. Que dividiu comigo as salas de aula do Pré-Vestibular São Francisco de Paula, na paróquia que leva o nome do mesmo santo, as disciplinas de Física por bons dois anos.

Conforto-me na certeza de que guiará meu caminho, de perto, no mundo das ciências que tanto discutíamos e amávamos.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por sempre ter me protegido e ter me feito curioso desde que tenho recordação.

Agradeço a minha paciente e dedicada orientadora Andressa dos Santos Nicolau, que se demonstrou uma verdadeira acadêmica, profissional da educação e psicóloga para conseguir lidar comigo.

Agradeço a minha amada e maravilhosa mãe, Major Elaine, que me deu estudo, nunca me deixou desistir e me abater. Com certeza a pessoa mais importante de todo essa história.

Agradeço a minha querida e amada irmã, Júlia, que sempre me fez “voltar à Terra depois de longas órbitas”.

Agradeço a minha maravilhosa e amada namorada, Renata Rauber Dahmer, sem ela esse trabalho não teria sido concluído dentro do prazo, e eu não conseguiria alcançar os outros objetivos que estava buscando alcançar.

Agradeço ao meu amado pai, que me acompanhou de perto nos piores momentos que tive na universidade.

Agradeço em especial, a minha amada avó Célia, por ter sido mais rigorosa nos estudos do que minha própria mãe, por ter me criado como filho, e por ter me vigiado como uma águia vigia seu ninho.

Agradeço em especial a meu avô Estanislau, por ter discordado de quase tudo que falei na vida, e mesmo assim ter me criado como filho e me amado incondicionalmente. Além do grandioso exemplo de vida, que me inspira a cada dia.

Agradeço a minha amada tia Cintia, que sempre acreditou em mim, desde o dia que nasci, e por ter me dado afilhadas maravilhosas e um grande tio (Dentinho).

Agradeço ao meu amado padrinho Alexandre, que sempre nos ajudou em tudo e sempre foi um grande exemplo de homem para mim.

Agradeço a minha vozinha Júlia, pelos paparicos e pelas massagens, tão importantes em dias de estresse comuns a um universitário.

Agradeço em especial ao meu avô Antônio, pelos conselhos, pela bondade, pela hombridade, pela honestidade, pelas risadas, pelo sarcasmo e pelo grande exemplo de vida que é para mim.

Agradeço, de forma muito especial, ao Laboratório de Super-Espectroscopia do Rio, o grandioso LASER, ao meu eterno e nada trivial orientador Cláudio Lenz, ao meu grande amigo e guru Álvaro Oliveira, o paciente e didático Rodrigo Nascimento, ao cearense mais carioca Levi Oliveira e à rainha de todos nós Wania Wolf, que se

tornaram uma segunda família para mim em nossas empreitadas pelo mundo dos lasers e átomos.

Agradeço ao meu grande amigo Luís Paulo, por não perder o bom humor nunca e sempre estar ao meu lado.

Agradeço ao meu grande amigo Lucas Paiva, pelo apoio e ombro amigo que nunca me foi negado.

Agradeço ao meu grande amigo Luiz Felipe Waitz, pelos conselhos e discussões.

Agradeço a Maria Lúcia e Baiana, pela ajuda em casa, conversas e darem suporte imprescindível para que isso fosse possível.

Resumo do Projeto de Graduação apresentado à Escola Politécnica/UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro nuclear.

Lucas da Silva Moreira

Janeiro/2019

Orientadora: Profª. Andressa dos Santos Nicolau

Curso: Engenharia nuclear

Uma usina nuclear monitora simultaneamente grande quantidade de parâmetros a fim de permitir que os operadores tenham uma vasta gama de informações sobre a condição da usina, porém, o ser humano é limitado cognitivamente e, muitas vezes, pode fazer análises erradas dos parâmetros apresentados, e em condições de estresse essa capacidade cognitiva se reduz de forma ainda mais acentuada. Sendo assim, este trabalho tem como objetivo criar um método de classificação baseado aprendizado de máquina (redes neurais) para classificar a condição de operação normal da usina bem como condições de acidentes, como o acidente por perda de refrigerante (LOCA), a ruptura de tubos do gerador de vapor (SGTR) e o blackout da estação, através de um conjunto mínimo de variáveis de estado julgadas necessárias para a classificação do evento em curso. Através da linguagem de programação Python e da biblioteca TensorFlow foram programadas as Redes neurais, os programas de teste de topologias e teste de combinações de variáveis de estado, para encontrar as topologias e combinações de variáveis de estado que otimizassem a precisão das classificações das redes. Os resultados mostram a eficiência do método proposto, o qual foi capaz de apresentar resultados satisfatórios para o problema de identificação de uma usina do tipo PWR, com apenas 4 variáveis de estado.

Palavras-chaves: Redes Neurais; Aprendizado de Máquina; Acidentes em Centrais Nucleares; Classificação de Acidentes.

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of nuclear Engineer.

Lucas da Silva Moreira

January/2019

Advisor: Andressa dos Santos Nicolau

Course: nuclear Engineering

A nuclear power plant simultaneously monitors a large number of parameters in order to allow operators to have a wide range of information about it's condition, but the human being is cognitively limited and can often make erroneous analyzes of the parameters presented, and under stress conditions this cognitive ability is reduced even more sharply. This work aims to create a classification method based on machine learning (neural networks) to classify the normal operating condition of the plant as well as accident conditions, such as the accident caused by loss of coolant (LOCA), steam generator tube rupture (SGTR) and station blackout, through a minimum set of state variables judged necessary for the classification of the current event. Through the Python programming language and the TensorFlow library, neural networks, topology test programs and state variable combinations tests were programmed to find the topologies and combinations of state variables that optimize the accuracy of the classifications of the networks. The results show the efficiency of the proposed method, which was able to present satisfactory results for the identification problem of a PWR type plant with only 4 state variables.

Keywords: Neural Networks; Machine Learning; Nuclear Power Plants Accidents; Classification of Accidents.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

CAPÍTULO 1
INTRODUÇÃO

O cuidado com a segurança de centrais nucleares é muito rigoroso, pois, dependendo do tipo de acidente, pode ocorrer derretimento do núcleo do reator, vazamento de material radioativo para o lençol freático, liberação de material radioativo na atmosfera, etc, afetando, assim, toda a região circundante à planta e gerando grandes impactos ambientais, econômicos e sociais.

No entanto, os acidentes evoluem de diferentes maneiras e necessitam de uma tomada de decisão imediata por parte dos operadores da sala de controle, de modo a evitar/mitigar danos à planta e à região circundante. Porém, por mais que os operadores sejam arduamente treinados, é sabido que o ser humano tem sua capacidade de cognição limitada em situações de estresse, o que se aplica ao caso de acidentes em plantas nucleares, principalmente por apresentar simultaneamente diversas variáveis de estado, que refletem o estado de diferentes processos da planta.

Dessa forma, é de grande relevância o desenvolvimento de métodos que sejam capazes de identificar o evento em curso (acidente) de modo a auxiliar o operador na tomada de decisão em um intervalo de tempo menor e de forma precisa, aumentando o nível de segurança da planta e diminuindo a carga cognitiva do operador.

Tendo em vista a necessidade de identificação rápida de um acidente, as limitações do ser humano de processar muitos parâmetros simultaneamente e tomar decisões em situações de alto estresse, ao longo dos anos têm sido proposto protótipos de computação baseados em inteligência artificial para suporte ao operador, que visam minimizar os erros cometidos pelos mesmos ao lidar com situações anormais da planta.

O uso de inteligência artificial na identificação de acidentes em instalações nucleares foi proposto por Eric B. Bartlett e Robert E. Uhrig (1991) (BARTLET e UHRIG, 1991) em seu trabalho "*nuclear Power Plant Diagnostics Using an Artificial Neural Network*", onde usaram redes neurais artificiais para o diagnóstico de falhas. Em 1994, Eric B. Bartlett e Anujit Basu (BASU e BARTLETT, 1993) aprimoram o trabalho de Bartlett e Uhrig, utilizando uma arquitetura composta de duas redes neurais artificiais que identificavam 27 transientes de um reator BWR, usando 97

variáveis de processo da planta. Esses trabalhos, impulsionaram uma gama de outros tantos que viriam a mostrar os benefícios da utilização da inteligência artificial na identificação de acidentes em centrais nucleares (BARTAL, LIN e UHRIG, 1995; FURUKAWA, UEDA e KITAMURA, 1995; JEONG, FURUTA e KONDO, 1996; MOL, 2002; MEDEIROS,2005; HSIAO, LIN e YUANN, 2010; VILLAS BÔAS, 2011; NICOLAU, 2014).

O conceito de redes neurais (HAYKIN, 2001) vem da área da ciência da computação que mais cresce nos dias atuais, a área de inteligência artificial. Inspiradas na biologia do cérebro humano, as redes neurais são usadas para identificar e aprender com padrões que lhes são apresentados e a partir disto, gerar resultados tanto preditivos (ex:preço de um ativo futuro baseado no seu custo ao longo dos anos) quanto classificativos (ex:determinar se uma mulher está ou não grávida de acordo com as imagem do seu útero, baseada no treinamento um número considerado de imagens de úteros com ou sem fetos).

Uma rede neural é um mecanismo de aprendizado de máquina (Machine Learning) [FAUSETT, 1994].A área de redes neurais vem se desenvolvendo desde a década de 40 do século passado, quando cientistas formularam a primeira representação matemática do neurônio, dando material para Frank Rosenblatt criar o Perceptron (ROSENBLATT, 1957), considerado o primeiro classificador a imitar o comportamento neuronal. Anos mais tarde, A. Ivakhnenko e V.G Lapa propuseram a utilização de mais neurônios simultaneamente em problemas classificatórios, como o cérebro, de fato funciona, permitindo assim a resolução de problemas mais complexos, e abrindo o caminho para o futuro desenvolvimento de redes de alto desempenho classificatório como o "*Cognitron*" criado por K. Fukushima (FUKUSHIMA, 1975) e mais tarde a criação da biblioteca TersorFlow pelo Google (ABADI *et al.*, 2016).

A biblioteca TensorFlow se utiliza de grafos e operações tensoriais para facilitar a implementação e a computação dos resultados, além de permitir o uso de processamento multicamadas devido ao uso da GPU do computador, aumentando muito o desempenho em relação a outros métodos. Além disso, a biblioteca é de código livre e vem com diversos métodos de aprendizagem e funções de ativação.

Neste contexto, o presente trabalho propõe uma metodologia para classificação de acidentes de base de projeto da usina nuclear Angra 2 através da metodologia redes neurais, usando a linguagem de programação Python em conjunto com a biblioteca de aprendizado de máquina e cálculo tensorial TensorFlow. A metodologia desenvolvida utiliza de dados das evoluções temporais de dezessete variáveis de estado de três situações de acidentes e a condição de operação normal da usina, onde busca encontrar a melhor classificação com o menor conjunto de variáveis possível.

O capítulo 2 deste trabalho apresenta uma descrição do problema de identificação de acidentes e os acidentes de base de projeto de uma usina do tipo PWR abordados neste trabalho.

O capítulo 3 apresenta uma breve descrição de redes neurais.

O capítulo 4 descreve e explica a metodologia desenvolvida para o problema de acidentes e os resultados obtidos.

No capítulo 5 são apresentados os resultados obtidos com a metodologia desenvolvida.

Finalmente, o capítulo 6 apresenta as conclusões desse trabalho.

CAPÍTULO 2
O PROBLEMA DA IDENTIFICAÇÃO DE ACIDENTES

Uma central nuclear é composta de uma grande variedade de sistemas e equipamentos, que seguem uma filosofia de segurança onde são projetados, construídos e operados com os mais elevados padrões de qualidade e confiabilidade. Assim, a operação de uma central, envolve o controle de centenas de variáveis pelos sistemas, de forma que a usina possa identificar, diagnosticar e corrigir falhas a tempo, para que a integridade da planta seja preservada (NICOLAU, 2014).

As atividades de monitoração, identificação e tomada de decisão no caso de eventos anormais são iniciadas pelos operadores na sala de controle. O desempenho dos operadores em uma situação de emergência, se dá quase em sua totalidade, na capacidade dos operadores em identificar corretamente o evento em curso e tomar as ações corretivas com a máxima certeza associada no menor espaço de tempo possível. Porém, nestas condições adversas a realização dessas atividades se torna cada vez mais complexa por diversos fatores (NICOLAU, 2014).

Em uma central nuclear, um desses fatores está na gama de atividades relacionadas com as atividades de identificação e diagnóstico de anomalias, tais como falhas nas unidades de processo, degradação dos processos da unidade, desvios de parâmetros das variáveis de processo, entre outros. Desta forma, na ocorrência de um evento anormal, as atividades de identificação e diagnóstico para a posterior tomada de decisão se tornam mais complexas ainda devido ao grande número de informações oriundas de instrumentos de medidas e alarmes inerentes da planta. As medidas realizadas pela instrumentação da planta podem, em alguns casos, serem incompletas ou não confiáveis devido a fatores como, por exemplo, imprecisões, desvios ou falhas em algum instrumento de medida, o que dificulta de forma mais significativa a condição dos operadores que, mesmo com árduo treinamento em diversos cenários de condições da planta, são postos em condições de extrema quantidade de informações e estresse (NICOLAU, 2014), diminuindo, assim, suas capacidades de cognição e aumentando as probabilidades de falha humana. (ROTH e MUMAW; HWANG e LIN, 1999).

O problema de identificação de acidentes é crucial na área nuclear, que está diretamente relacionado à segurança da usina. Quando ocorre um transiente, é

possível observar a evolução temporal das variáveis da planta envolvidas, através de leituras de instrumentos de monitoramento. A evolução de cada variável apresenta uma curva definida (padrão). Esses padrões são únicos em relação ao tipo de acidente. Esse padrão pode ser usado para identificar o acidente. Assim, o o problema de identificação de acidentes pode ser visto como um problema de reconhecimento de padrões, onde sintomas relevantes compostos por um conjunto de variáveis das plantas são identificados como representativos do acidente.

Ao longo dos anos, alguns métodos estatísticos e de inteligência artificial foram usados como ferramentas de base para sistemas protótipos de identificação de acidentes em usinas nucleares (KWON e KIM, 1999; MOL *et al.*, 2002; MEDEIROS, 2005; VILLAS BÔAS, 2011; NICOLAU *et al.*, 2017; PENG *et al.*, 2018; PINHEIRO, 2019), onde dados simulados de acidentes de base de projeto são usados como padrões de acidentes a serem identificados pela ferramenta desenvolvida.

Apesar das várias pesquisas, até o nosso conhecimento, não se tem no mercado nuclear uma ferramenta capaz de classificar com precisão um evento anormal em curso, e dessa forma auxiliar o operador em sua tomada de decisão, diminuindo sua carga cognitiva e com isso aumentando ainda mais o nível de segurança da usina.

2.1. Análise de acidentes

A indústria nuclear é amplamente reconhecida no mundo pela alta preocupação com a segurança de suas instalações. Desde a concepção do projeto até a sua conclusão, todos os detalhes que dizem respeito à segurança da instalação e da população no entorno da planta são analisados e estudados minuciosamente, para que os riscos provenientes do empreendimento sejam reduzidos o quanto possível.

Para reduzir esses riscos, são utilizadas analises determinísticas e probabilísticas, a fim de determinar os efeitos de cada tipo de acidente no que concerne a própria planta e ao meio ambiente.

A análise determinística consiste em determinar por meio de simulações, cálculos, e procedimentos dos mais variados se uma instalação é tolerante a acidentes de base de projeto (DBA), que são acidentes postulados em uma planta nuclear, em

que seus equipamentos, sistemas e estruturas devem estar preparados para evitar liberação de material radioativo ao público e meio ambiente caso os mesmo venham a ocorre. Por meio destas analises, pode-se estabelecer os limites de operação segura de uma central nuclear (PETRANGELI, 2006; KHATTAK *et al.*, 2018).

A análise probabilística consiste em uma série de métodos de análise que fornecem de forma abrangente e estruturada maneiras de identificar cenários de possíveis falhas e obter estimativas numéricas dos riscos da central nuclear. O modelo de analise probabilística fornece uma abordagem sistemática para determinar se os sistemas de segurança são adequados, se o projeto da planta está de acordo com o requerido pelos órgãos reguladores, se o conceito de defesa em profundidade foi aplicado e se o risco é tão baixo quanto possível (SEHGAL, 2012).

Os acidentes que a partir das análises tem uma maior relevância e maior probabilidade de ocorrência são usados como base para a concepção de todas as estruturas, sistemas e componentes da instalação nuclear, em especial os sistemas de segurança (LITHUANIAN ENERGY INSTITUTE, 1997; PETRANGELI, 2006).

A partir desse rigoroso planejamento das estruturas e sistemas, quando concluída e operável, a usina nuclear deve ser munida de sistemas de segurança capazes de, dada uma determinada situação fora da normalidade, reestabelecer as condições normais de operação, ou em caso de acidentes severos, evitar liberação significativa de material radioativo ao ambiente externo (LITHUANIAN ENERGY INSTITUTE, 1997).

Desta forma, a indústria nuclear se utiliza do conceito largamente conhecido dentro da filosofia de segurança da indústria nuclear, que é o conceito de defesa em profundidade (Segurança multi nível), que é caracterizado como o estudo da evolução de um DBA, e a implantação de sistemas e estruturas que possam proteger a planta de um eventual acidente (SEHGAL, 2012).

Portanto, os sistemas e estruturas irão ser preparados para lidar com diversos tipos de DBA, tais como (PETRANGELI, 2006):

- Abertura espúria de uma das válvulas de segurança do pressurizador;
- Perda de energia de todas as bombas do primário;
- Ruptura em um dos tubos do gerador de vapor (SGTR);

- Ruptura da tubulação de maior diâmetro (LOCA);
- Ejeção de uma barra de controle do núcleo.

Tudo previamente explicitado, é parte de um conceito mais amplo denominado cultura de segurança (PETRANGELI, 2006), o qual, em essência, está ligada aos meios pelos quais a atenção estrita à segurança é obtida tanto no campo organizacional como no individual. A cultura de segurança do ponto de vista de uma central nuclear tem como principais aspectos:

- Consciência individual dos funcionários da central nuclear sobre a importância da segurança;
- Treinamento de pessoal por iniciativa da própria central nuclear;
- Demonstração da gerência em nível sênior da alta prioridade da segurança;
- Sistemas de premiações através de atitudes geradas pelos próprios indivíduos;
- Supervisão e auditoria;
- Responsabilidade, através da atribuição formal e da descrição de deveres e da sua compreensão pelas pessoas.

Desta forma, o objetivo da análise de segurança é o de estabelecer e confirmar, através de ferramentas analíticas, a base de projeto para os equipamentos e sistemas de segurança, além de assegurar que o projeto da planta seja capaz de atender aos limites prescritos e aceitáveis para as liberações e doses de radiação para cada condição operacional da usina.

Neste trabalho foram usados como base de análise da metodologia proposta 3 acidentes de base de projeto postulados para uma usina nuclear do tipo PWR: acidente por perda de refrigerante (LOCA), Ruptura dos tubos do gerador de vapor (SGTR) e blackout da estação. O uso do LOCA e SGTR se da pelo fato dos dois terem uma similaridade grande quando se trata de suas respectivas evoluções temporais, podendo em um cenário real confundir o operador pela sutil diferença que separa um do outro, desta forma, desenvolver um sistema que caracterize tanto o acidente LOCA quanto SGTR testa tanto eficácia da metodologia quanto permite ajudar iniciar um projeto que futuramente poderá ajudar operadores na identificação dos acidentes com maior taxa de confusão. O uso do acidente Blackout se dá devido as suas características bem particulares, se diferenciando completamente dos outros em sua evolução temporal. E o uso da operação normal ocorre para se ter o potencial de

identificação e monitoração da usina em sua condição padrão, para assim ser possível identificar algum desvio da normalidade. Os tipos de acidentes foram escolhidos para efeitos de comparação dos resultados encontrados com os disponíveis na literatura.

2.1.2. Acidente por perda de refrigerante (LOCA)

O acidente de base de projeto exigido pela *nuclear Regulatory Commission* (NRC) para testar projetos de plantas nucleares é o acidente por perda de refrigerante (LOCA), que, por sua vez, pode ocorrer nas mais variadas escalas, que são diretamente relacionadas com o tamanho da ruptura ocorrida no circuito primário, que culmina no vazamento do refrigerante (SEHGAL, 2012; RAGHEB, 2016).

O processo pelo qual um LOCA pode ocasionar um dano ao núcleo é simples, conforme a vazão de refrigerante decresce, com o tempo, devido ao vazamento, a taxa de troca de calor com o circuito secundário também cai, fazendo com que o elemento combustível não seja resfriado como o usual e, por sua vez, aumente a sua temperatura. Mesmo com o desligamento do reator após a detecção da queda de pressão no circuito primário, o calor de decaimento pode danificar o núcleo, e assim, os sistemas de segurança relacionados ao arrefecimento precisam atuar (RAGHEB, 2016; PETRANGELI, 2006).

O aumento de temperatura causa uma série de fenômenos físicos no núcleo do reator, apresentados na Tabela 1 abaixo (RAGHEB, 2016):

Tabela 1 – Fenômenos físicos em função da temperatura.

Fenômeno Físico	***Temperatura [C°]***
Temperatura média do revestimento durante a operação normal.	350
O revestimento é perfurado ou incha devido ao acúmulo de pressões internas. Alguns gases de fissão, como Kr, I, e Xe são liberados; A reação entre aço inoxidável e o Zircaloy se inicia. O inchaço do revestimento pode impedir o fluxo de refrigerante em alguns canais.	800-1450
O vapor pode reagir com o revestimento de Zircaloy produzindo hidrogênio e uma liberação de energia que excede a do calor de decaimento. O processo de oxidação fragiliza o Zircaloy. As ligas de aço se fundem.	1450-1500
A reação de Zircaloy e vapor pode se tornar autocatalítica, alimentando-se a si mesma, a menos que o Zircaloy seja resfriado por imersão no refrigerante	1550-1650
O revestimento de Zircaloy se funde. A liberação de produtos de fissão vindos da cerâmica de UO2 do combustível se torna significativa acima de 2150K.	1900
Ambos UO_2 e ZrO_2 se fundem.	2700

Fonte: Ragheb (2016).

Dessa forma, é possível concluir que, se o acidente por perda de refrigerante não for adequadamente detectado e mitigado pelos sistemas de segurança, pode causar graves danos ao núcleo.

2.1.3. Ruptura de tubos do gerador de vapor (SGTR)

O acidente por ruptura dos tubos do gerador de vapor consiste em um vazamento do refrigerante do circuito primário para o secundário. Assim como o acidente por perda de refrigerante, em que se pode observar que o refrigerante é liberado fora do sistema de resfriamento do reator, caracterizando um bypass do circuito primário para a contenção.

No SGTR, como há um vazamento de refrigerante do circuito primário para o secundário, existe uma queda no potencial de resfriamento do núcleo e uma contaminação radiológica do circuito secundário, que por sua vez pode resultar em uma liberação de produtos de fissão ao ambiente externo (MACDONALD, SHAH, *et al.*, 1996; LEE, 2003).

Ao se ter indicativos de um SGTR, as ações que devem ser tomadas para encerrar o acidente são: providenciar um resfriamento apropriado do núcleo; e, equilibrar as pressões do primário e secundário, já que é a diferença de pressão que faz o fluido refrigerante, contaminado por produtos de fissão, fluir do primário para o secundário (MACDONALD, SHAH, *et al.*, 1996).

2.1.4 Blackout

Grande parte dos equipamentos de uma instalação nuclear, principalmente os das usinas nucleares PWR brasileiras, Angra 1 e Angra 2, são alimentados por energia elétrica em forma de corrente alternada, energia essa que advém da rede externa. No caso de haver uma falta de energia, a planta tem métodos que consistem de ligações a redes externas distintas e geradores de emergência para suprir a necessidade de corrente alternada da usina caso haja uma queda na rede principal de alimentação (IAEA, 1985; MAZZONI, 2018).

Sabendo disso, o pior evento que se pode ocorrer é a perda da energia, tanto externa quanto interna da instalação, fazendo com que equipamentos essenciais para o funcionamento da usina e proteção do núcleo fiquem indisponíveis, denomina-se a esse evento blackout. Portanto, para ser licenciada, uma usina tem que ter métodos de resistir e se recuperar de um blackout (MAZZONI, 2018).

Dessa forma, um blackout só se dá quando há a perda de energia externa e interna da planta simultaneamente, no entanto não ocorre o comprometimento do sistema de proteção do reator, pois este fica responsável pelo seu desligamento, assim como mostra a árvore de eventos da Figura 1 (KOK, 2016):

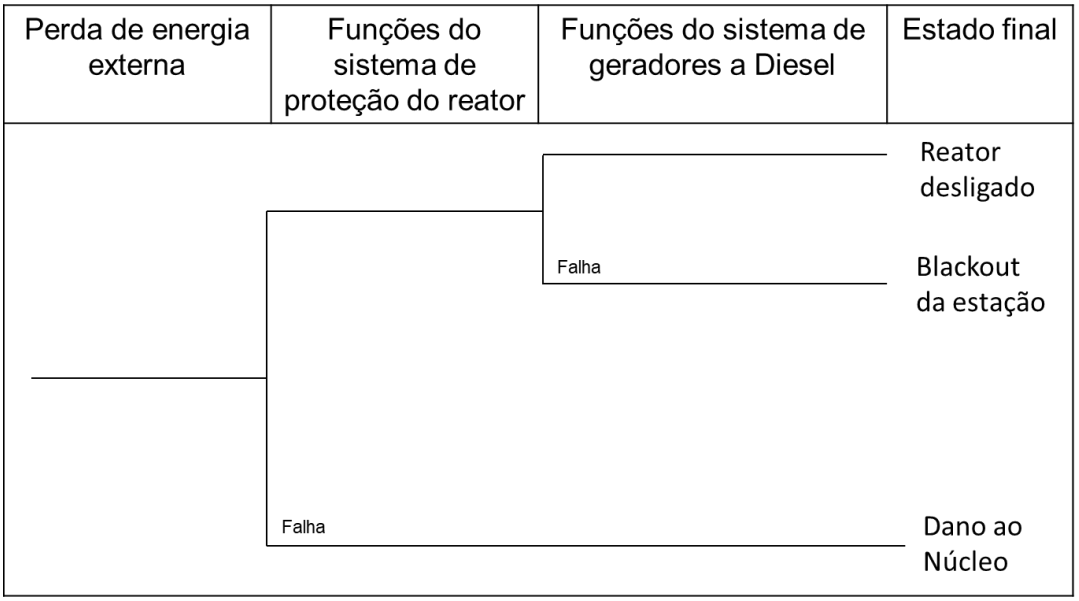

Figura 1 – Árvore de eventos. Fonte: Kok (2016).

CAPÍTULO 3
REDES NEURAIS

O cérebro é o órgão mais complexo do corpo humano, e o que mais atrai atenção de pesquisadores de diversas áreas pela sua capacidade de abstração, aprendizado e memorização. Desde a década de 40 do século passado, existe uma vontade crescente de emular a biologia cerebral, com suas sinapses, axônios e dendritos, na tentativa de aproveitar o que o cérebro humano tem de melhor para resolver problemas nas mais diversas áreas do conhecimento.

O psiquiatra e neuroatomista Warren Maculloch e o matemático Walter Pitts deram início a um grande campo de estudo dentro da área de inteligência artificial, ao publicar o trabalho *"A Logical Calculus of the Ideas Immanent in Nervous Activity"* em 1943 (MCCULLOCH e PITTS, 1943), no qual descrevem o cálculo lógico das redes neurais, de forma a unir os estudos neurofisiológicos e da lógica matemática.

Alguns poucos anos depois, o recém considerado maior cientista do século XX, Alan Turing, fez previsões sobre o aprendizado de máquina, o impacto deste sobre o empregos e desenvolveu um método de como um computador deveria "pensar" em seu artigo "*Computing Machinery and Intelligence*" (TURING, 1950). O método que recebeu de Turing o nome de "Jogo da Imitação" deu nome ao filme em sua homenagem.

Em 1957, Frank Rosenblatt submete seu artigo "*The Perceptron: A Perceiving and Recognizing Automaton*" (ROSENBLATT, 1957) ao Laboratório de Aeronáutica de Cornell, declarando ter criado um mecanismo de reconhecimento de padrões similar ao cérebro humano, dando um dos passos mais importantes para o desenvolvimento do aprendizado de máquina e das redes neurais.

Após 10 anos do grande passo de Frank Rosenblatt, A. Ivakhnenko e V.G Lapa propuseram e desenvolveram as primeiras redes neurais com múltiplas camadas em seu artigo "*Cybernetic Predicting Devices*", sendo considerados os pais do aprendizado de máquina profundo.

Porém, o grande salto do reconhecimento de padrões veio com Kunihiko Fukushima ao desenvolver uma rede neural hierárquica multicamadas capaz de um

robusto reconhecimento de padrões visuais por meio de aprendizado, chamada *Cognitron,* proposto por ele no seu artigo "*Cognitron: A self-organizing multilayered neural network*" (FUKUSHIMA, 1975).

Encontrar padrões é uma tendência natural da espécie humana, e já na segunda década do século XXI também uma tendência de grande parte dos dispositivos eletrônicos e máquinas. Com o crescimento das gigantes do Vale do Silício como Apple, Facebook e Google, a demanda por estudos e desenvolvimento na área de aprendizado de máquina cresceu exponencialmente, e em virtude disto, o avanço na área cresceu de forma semelhante. Devido a necessidade competitiva, em 2011, a equipe de pesquisa do Google chamada GoogleBrain (ABADI, *et al.*, 2016), desenvolveu internamente para fins comerciais e de pesquisa, a biblioteca de aprendizado de máquina "*DistBelief*". Com a popularidade dentro das empresas do Google, cientistas foram contratados para otimizar a biblioteca, e em 2015 foi lançado o TensorFlow, que é a biblioteca de aprendizado de máquina onde se pode encontrar grande parte dos mais recentes avanços em redes neurais (ex: funções de ativação que permitem treinar redes com desempenhos nunca antes alcançados, como as funções ReLU (HAHNLOSER *et al.*, 2000, ELU (CLEVERT, UNTERTHINER e HOCHREITER, 2016) e Softmax (BRIDLE, 1990)) de forma a ser uma ferramenta para que os programadores desenvolvessem aplicações cada vez mais complexas, treinar redes cada vez mais sofisticadas e com as metodologias que melhor se adaptassem aos seus problemas . Uma grande vantagem desta biblioteca é o fato da mesma ser de código aberto, permitindo o próprio usuário contribuir para a evolução da mesma (ABADI *et al.*, 2016).

Redes neurais é um sistema de processamento de informação, com o objetivo de recriar o potencial cerebral (FAUSETT, 1994; HAYKIN, 2001), através de abstrações matemáticas da cognição humana. Essas abstrações são realizadas a partir das seguintes suposições:

- O processo da informação ocorre em diversos elementos simples, chamados neurônios;
- Sinais são passados entre neurônios através de elos de conexão;
- Cada conexão tem um peso associado que em uma rede neural típica multiplica o sinal transmitido;
- Cada neurônio tem uma função de ativação que limita e adiciona não-linearidade aos seus valores de saída;

- Uma regra de propagação dos sinais;
- Uma regra de aprendizado (adaptação dos pesos).

Portanto, uma rede neural consiste de um determinado número de unidades de processamento chamadas neurônios, que são interligados uns com os outros através de suas conexões únicas devido aos pesos associados a cada conexão. Os pesos de cada conexão são uma característica da própria rede e se adaptam a cada problema

Na biologia, os pesos são influenciados pelas emoções, sentimentos e mais precisamente pela presença ou falta de serotonina. Por outro lado, o que no neurônio artificial é representado pelos pesos, na biologia é representado por reações químicas, e cabe a estas atenuar ou amplificar cada sinal elétrico que passa pelas sinapses a fim de fazer o ser humano tomar decisões.

3.1. Neurônio artificial

O neurônio é a unidade de processamento fundamental de uma rede neural. Na Figura 2 é apresentado o diagrama de um neurônio artificial usado na construção de redes neurais de diversas espécies.

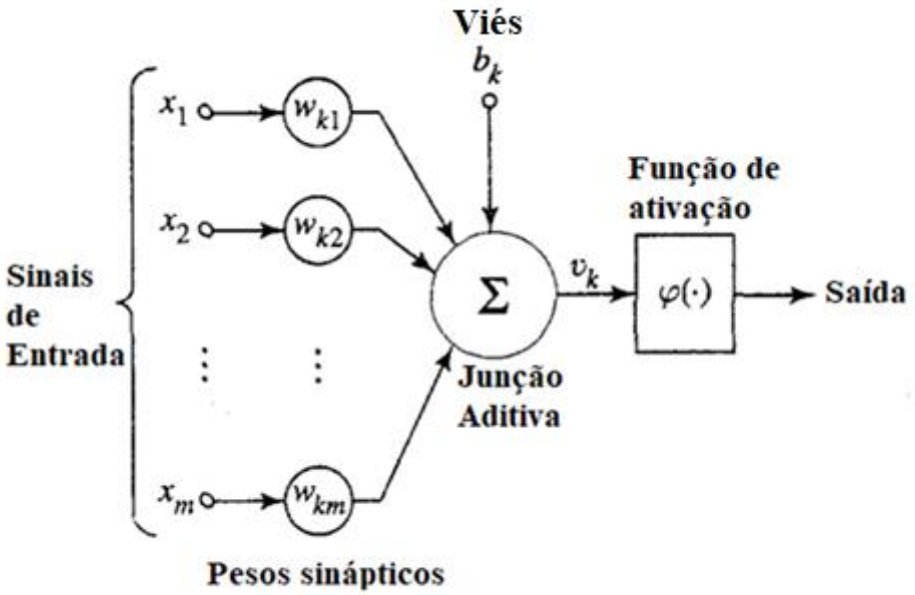

Figura 2 – Representação de um neurônio artificial. Fonte: Haykin (2001).

A partir dessa figura, podem-se identificar três elementos da abstração matemática do neurônio (HAYKIN, 2001):

1. Um conjunto de sinapses ou elos de conexão, cada uma com um peso próprio. Dessa forma, um determinado sinal de entrada x_j na entrada da sinapse *j* se conecta ao neurônio *k* é multiplicado pelo peso w_{kj}. O primeiro índice se refere ao próprio neurônio, enquanto o segundo se refere ao número do neurônio de onde o sinal está saindo;
2. Uma operação de somatório que irá somar os sinais de entrada ponderados pelos pesos sinápticos, ou seja, ocorre uma combinação linear dos sinais.
3. Uma função de ativação que tem o objetivo de restringir a amplitude do sinal de saída do neurônio a valores dentro de um limite tanto superior quanto inferior.

No modelo do neurônio artificial representado na Figura 2, também é incluído um viés (*bias)*, que serve para aumentar ou diminuir a entrada liquida da função de ativação (HAYKIN, 2001). Dessa forma, podemos representar o neurônio artificial por duas equações:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{1}$$

$$y_k = \varphi(u_k + b_k) \tag{2}$$

AEq. (1) pode ser escrita em sua forma vetorial, facilitando a implementação computacional:

$$u_k = \boldsymbol{w_k}.\boldsymbol{x} = < \begin{bmatrix} w_{k1} \\ w_{k2} \\ w_{k3} \\ \vdots \\ w_{km} \end{bmatrix}, \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_m \end{bmatrix} > = w_{k1}x_1 + w_{k2}x_2 + w_{k3}x_3 + \cdots + w_{km}x_m \tag{3}$$

As variáveis são definidas como (HAYKIN, 2001; MARTINS, 2011; FAUSETT, 1994):

- $\boldsymbol{w_k}$ é o vetor dos pesos sinápticos relacionados ao neurônio *k*, $\boldsymbol{w_k} = \begin{bmatrix} w_{k1} \\ w_{k2} \\ w_{k3} \\ \vdots \\ w_{km} \end{bmatrix}$;
- $\boldsymbol{w_{km}}$ é a representação para o peso sináptico do neurônio *k* com relação à entrada *m;*
- $\boldsymbol{x}$ é o vetor dos sinais de entrada relacionados ao neurônio *k*, $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_m \end{bmatrix}$;
- $\boldsymbol{x_m}$ é a representação para os sinais de entrada, onde *m* é o número da entrada;
- $\boldsymbol{u_k}$ é a saída da combinação linear dos produtos dos sinais de entrada pelos pesos sinápticos;
- $\boldsymbol{b_k}$ é o bias (viés);
- $\boldsymbol{\varphi}$ é a função de ativação;
- $\boldsymbol{y_k}$ é o sinal de saída do neurônio.

O *bias (viés)* b_k tem como intuito realizar uma transformação afim à saída do u_k, ou seja, faz-se uma translação da função mantendo seu paralelismo, como se pode observar na Figura 3 (HAYKIN, 2001; MARTINS, 2011).

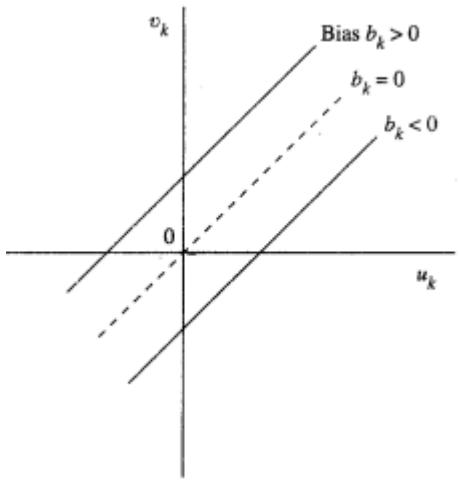

Figura 3 – Translação da função $v_k(u_k)$, devido ao *bias*. Fonte: Haykin (2001).

Também, pode ser definido o campo local induzido do neurônio *k* como v_k através da Eq. (4):

$$v_k = u_k + b_k \tag{4}$$

O bias é um fator externo ao neurônio *k*, mas com o uso das definições anteriores, podemos criar uma nova sinapse que o represente, de forma a torná-lo um parâmetro interno do neurônio (HAYKIN, 2001). Combinando as Eqs. (1), (4) e, adicionando uma nova sinapse ao neurônio, tem-se:

$$v_k = \sum_{j=0}^{m} w_{kj} x_j \tag{5}$$

Considera-se que, para formular matematicamente a adição de uma sinapse ao neurônio *k,* o índice do somatório passa a começar em $j = 0$, onde a entrada da sinapse deve ser $x_0 = +1$, de forma que o bias b_k seja o peso desta sinapse (HAYKIN, 2001):

$$w_{k0} = b_k \tag{6}$$

Substituindo a Eq. (4) na Eq. (2), e utilizando a consideração acima, tem-se:

$$y_k = \varphi(v_k) \tag{7}$$

Assim, os novos vetores dos pesos sinápticos e das entradas do neurônio *k* são alterados para:

$$\boldsymbol{w_k} = \begin{bmatrix} w_{k0} \\ w_{k1} \\ w_{k2} \\ w_{k3} \\ \vdots \\ w_{km} \end{bmatrix} \tag{8}$$

e

$$\boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_m \end{bmatrix} \tag{9}$$

A partir dessa nova configuração neuronal, podemos formular matematicamente a expressão v_k e demonstrar sua equivalência com a forma de cálculo proposta na Eq. (4). Portanto, ao representar $\boldsymbol{w_k}$ e $\boldsymbol{x}$ como se segue:

$$\boldsymbol{w_k} = \begin{bmatrix} w_{k0} \\ w_{k1} \\ w_{k2} \\ w_{k3} \\ \vdots \\ w_{km} \end{bmatrix} = \begin{bmatrix} b_k \\ w_{k1} \\ w_{k2} \\ w_{k3} \\ \vdots \\ w_{km} \end{bmatrix} \tag{10}$$

e

$$\boldsymbol{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} +1 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_m \end{bmatrix} \tag{11}$$

Logo, pode-se representar v_k segundo a Eq. (12), de forma semelhante a Eq. (4).

$$v_k = \boldsymbol{w_k}^T.\boldsymbol{x} = < \begin{bmatrix} b_k \\ w_{k1} \\ w_{k2} \\ w_{k3} \\ \vdots \\ w_{km} \end{bmatrix}, \begin{bmatrix} +1 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_m \end{bmatrix} > = b_k + w_{k1}x_1 + w_{k2}x_2 + w_{k3}x_3 + \cdots + w_{km}x_m = b_k + \sum_{j=1}^{m} w_{kj}x_j \tag{12}$$

Portanto, o campo local induzido v_k é igual em ambas as configurações neuronais, com bias b_k como parâmetro interno ou como parâmetro externo, como mostrado na Figura 4.

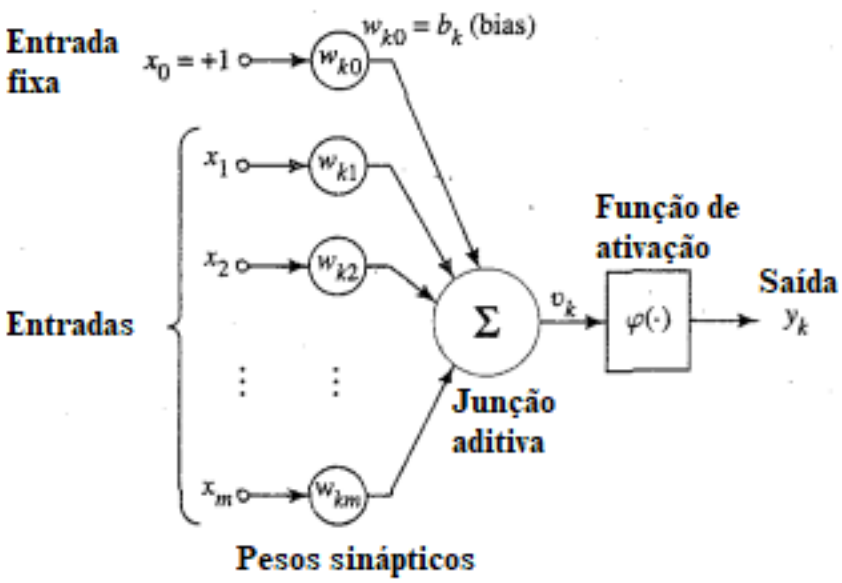

Figura 4 – Neurônio artificial com *bias* interno. Fonte: Haykin (2001).

3.2. Função de Ativação

A função de ativação define a saída de um neurônio quando aplicada sobre a combinação linear dos pesos sinápticos multiplicados pelas entradas. Essa saída é usada como entrada para outros neurônios na camada subsequente até que a solução desejada para o problema seja encontrada (HAYKIN, 2001).

As funções de ativação são elementos cruciais na arquitetura de uma rede neural, pois introduzem propriedades não-lineares na rede. Isso permite que a rede consiga aprender padrões para tipos não triviais de dispersão dos dados. Caso não existissem essas funções, a ativação de um neurônio artificial só poderia ser linear, aumentando assim o tempo de processamento (FAUSETT, 1994).

3.2.1. Função de Limiar

No modelo neuronal que se utiliza da função de limiar, a saída de um neurônio assume o valor 1 se o v_k é maior ou igual a 0 e, tem saída nula se v_k é menor que 0, definida como função de limiar, Eq. (13). Este modelo neuronal é chamado de

neurônio de McCulloch-Pitts (HAYKIN, 2001), em que, v_k é o campo induzido do neurônio, representado na Fig. 5.

$$y_k = \begin{cases} 1, se\ v_k \geq 0 \\ 0, se\ v_k < 0 \end{cases} \quad (13)$$

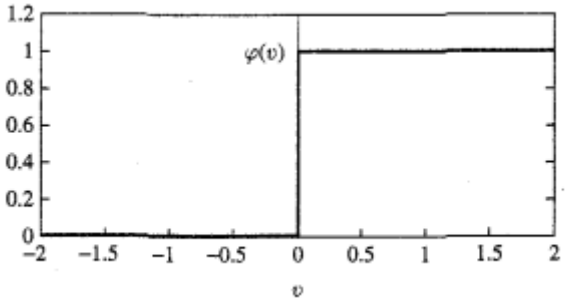

Figura 5 – Gráfico da função limiar. Fonte: Haykin (2001).

3.2.2. Função Logística (Sigmoide)

A função sigmoide é a função de ativação mais conhecida e utilizada em redes neurais de alimentação para frente (feedfoward), devido a diversas características, como (HAN, 1995; HAYKIN, 2001):

- Seu suficiente grau de suavidade;
- Sua não-linearidade;
- A simplicidade computacional da sua derivada;
- Comportamento balanceado entre linearidade e não-linearidade;
- Ser estritamente crescente;

A função sigmoide, é definida como:

$$\varphi(v_k) = \frac{1}{1 + e^{av_k}} \quad (14)$$

onde, a é o parâmetro que determina a inclinação da função. Se o limite de $\varphi(v_k)$ for tomado quando a tende a infinito, pode-se observar que a função sigmoide se torna uma função limiar (HAYKIN, 2001):

$$\varphi(v_k) = \lim_{a\to\infty}\left(\frac{1}{1+e^{av_k}}\right) = \begin{cases}1, se\ v_k \geq 0\\ 0, se\ v_k < 0\end{cases} \quad (15)$$

Pode-se observar na Fig. 6 o comportamento da função sigmoide, com a variação do parâmetro a.

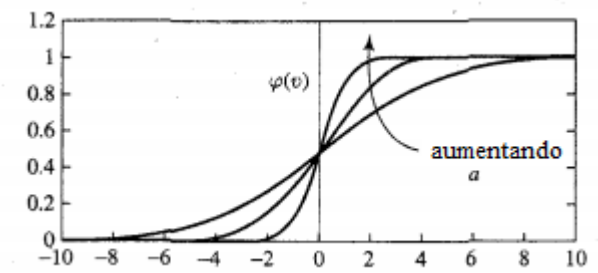

Figura 6- Variação da função Sigmóide com o parâmetro a. Fonte: Haykin (2011).

A derivada da função sigmoide é dada por:

$$\varphi'(v_k) = \varphi(v_k)(1-\varphi(v_k)) \quad (16)$$

3.2.3. Função Tangente Hiperbólica

A função tangente hiperbólica foi proposta devido à limitação da função sigmoide em lidar com valores negativos, pois a mesma torna os dados de saída relativos as entradas negativa muito próximos de zero, prejudicando a atualização dos parâmetros da rede neural, já a função tangente hiperbólica transforma o domínio do campo local induzido de (−∞, +∞) para (−1, +1) na saída da função (FARHADI, 2017). A função tangente hiperbólica tem a seguinte formulação matemática:

$$\varphi(v_k) = \frac{e^{v_k} - e^{-v_k}}{e^{v_k} + e^{-v_k}} \quad (17)$$

A derivada da função tangente hiperbólica é dada por:

$$\varphi'(v_k) = 1 - \varphi(v_k)^2 \quad (18)$$

A Figura 7 corresponde a sua derivada.

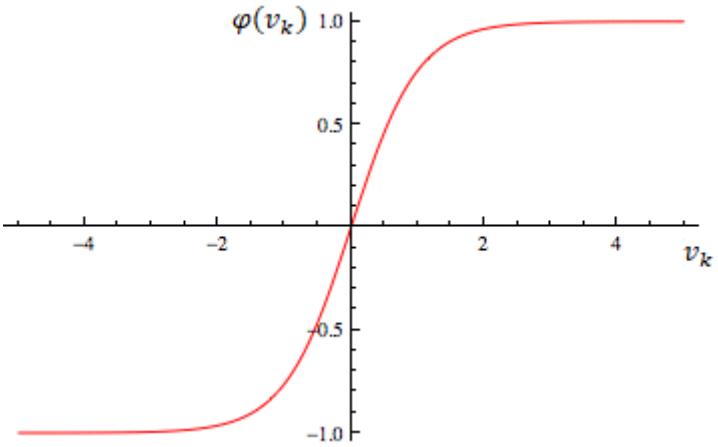

Figura 7- Gráfico da função tangente hiperbólica. Fonte: Weisstein (s/d).

3.2.4. Função Unidade Linear Retificada (ReLU)

Os neurônios logísticos são biologicamente mais plausíveis do que os neurônios tangentes hiperbólicos, tendo estes últimos um funcionamento melhor para o treinamento de redes de multicamadas. Em 2000, Hahnloser et al. (2000) propuseram em seu trabalho "*Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit"* a função de ativação ReLU, de forma que os neurônios que usam esta função de ativação representam um modelo ainda melhor quando se considera a similaridade com a biologia neuronal e, ainda possuem um desempenho igual ou superior aos neurônios tangentes hiperbólicos, apesar de sua não-linearidade e sua não-diferenciabilidade em zero (HAHNLOSER *et al.*, 2000).

A função ReLU retorna o valor zero para valores de entrada menores que zero e tem um aspecto linearmente crescente para valores maiores ou iguais a zero. Dessa forma, a função ReLU pode ser definida como:

$$y_k(v_k) = \begin{cases} 0 \; se \; v_k < 0 \\ v_k se \; v_k \geq 0 \end{cases} \tag{19}$$

E sua derivada é definida como:

$$y'_k(v_k) = \begin{cases} 0 \; se \; v_k < 0 \\ 1 \; se \; v_k \geq 0 \end{cases} \tag{20}$$

Porém, como a função ReLU retorna zero para qualquer valor de v_k abaixo de zero, a função tende a perder informação quando o conjunto de dados se estende

para valores negativos de v_k; dessa forma a função ReLU paramétrica foi proposta com a intenção de minimizar as perdas de informação devido a valores negativos de v_k, não mais retornando zero para valores de v_k negativos. Pode-se definir a função ReLU paramétrica como:

$$y_k(v_k) = \begin{cases} \alpha v_k \; se \; v_k < 0 \\ v_k se \; v_k \geq 0 \end{cases} \qquad (21)$$

E sua derivada é definida como:

$$y'_k(v_k) = \begin{cases} \alpha \; se \; v_k < 0 \\ 1 \; se \; v_k \geq 0 \end{cases} \qquad (22)$$

Pode-se analisar na Figura 8 o gráfico das funções definidas nas Eqs (19) e (21).

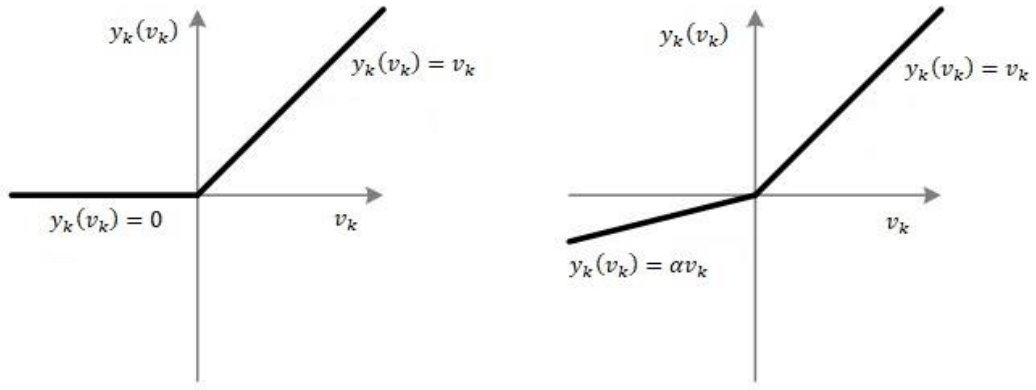

Figura 8 – Função ReLU (esquerda) e Função ReLU paramétrica (direita). Fonte: Sharma (2017).

3.2.5. Função Softmax

Em 1990, John S. Bridle propôs em seu trabalho ”*Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition*“, a função de ativação *Softmax,* que é uma generalização da função Sigmoide largamente utilizada na classificação em classes múltiplas mutualmente exclusivas (número de classes maior que 2) (BRIDLE, 1990). A função *Softmax* transforma as saídas para cada classe para valores entre 0 e 1 e então divide pelas

soma das saídas. Isso essencialmente dá a possibilidade de a entrada estar em uma determinada classe. A função sigmoide é capaz de lidar com apenas duas classes.

Essa função recebe um vetor de entradas e retorna um vetor com a probabilidade do pertencimento desse conjunto de entradas a cada classe, como é mostrado na Eq. (23)

$$y_i(\boldsymbol{x}) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}} = \begin{bmatrix} P(y_1) \\ P(y_2) \\ P(y_3) \\ \vdots \\ P(y_i) \end{bmatrix} \tag{23}$$

Assim como a derivada parcial da saída *i* em relação a entrada *j* é dada por (BENDERSKY, 2016):

$$\frac{\partial y_i(\boldsymbol{x})}{\partial x_j} = \begin{cases} y_i(\boldsymbol{x})\left(1 - y_j(\boldsymbol{x})\right) se\ i = j \\ -y_j(\boldsymbol{x}) y_i(\boldsymbol{x})\ se\ i \neq j \end{cases} \tag{24}$$

Por exemplo, no caso em que se tem como saída [1.2, 0.9, 0.75], quando aplica-se a função *Softmax*, tem-se [0.42, 0.31, 0.27] e isso pode ser usado como probabilidade de que o valor seja de cada classe.

Dessa forma, pode-se observar que a função *Softmax* deve ser aplicada em todas as saídas da camada anterior devido ao seu denominador, o que difere de todas as funções de ativação até aqui apresentadas, as quais devem ser aplicadas em cada elemento de entrada por vez, gerando, assim, uma única saída para cada elemento de entrada. Pode-se ver o funcionamento da função *Softmax* na Figura 9.

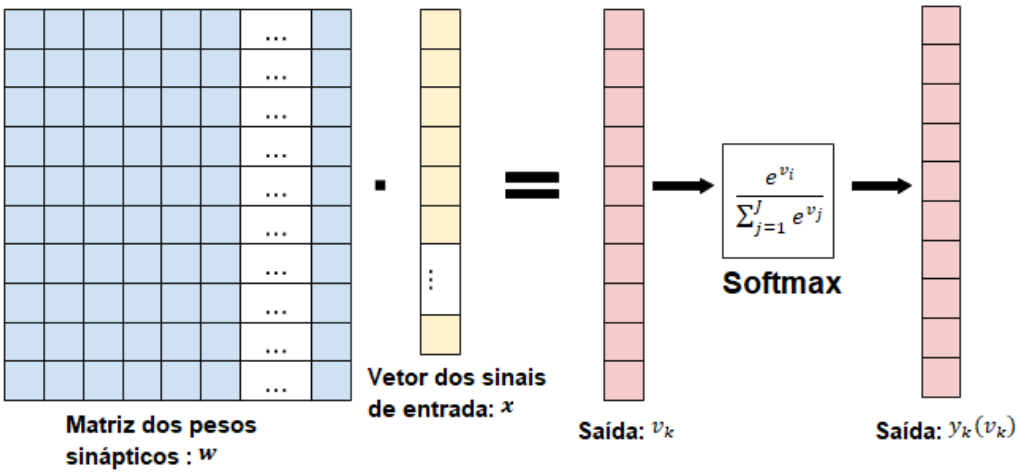

Figura 9 – Aplicação da função *Softmax.* Fonte: Berdersky (2016).

3.3. Redes neurais multicamadas

As redes neurais de múltiplas camadas consistem de uma camada de entrada, uma ou mais camadas ocultas de neurônios e uma camada de saída. O sinal de entrada de uma rede neural multicamadas se propaga através da rede, camada por camada até chegar à saída (HAYKIN, 2001). Essas redes são normalmente consideradas generalizações do *Perceptron* de Rosenblatt (1957), que topologicamente é um neurônio artificial cuja função de ativação é a função limiar.

Como o Perceptron de Rosenblatt consegue ser eficiente apenas para os casos em que os dados se encontram linearmente separáveis, essa abstração de multicamadas permite que a rede, através de algoritmos de aprendizado, consiga fazer a classificação em dados não linearmente separáveis, sendo, assim, muito utilizada nos dias de hoje para resolver diversos problemas complexos.

A propagação do sinal de entrada ocorre de forma semelhante tanto para um neurônio, como descrito na Seção 3.1, quanto para uma rede de múltiplas camadas, onde a saída de cada camada será a entrada da próxima, como se pode analisar no modelo de algoritmo de alimentação de avanço apresentado a seguir:

1. O campo local induzido da camada c se dá por: $\boldsymbol{v_{ck}} = \boldsymbol{w^{(c)^T}}.\boldsymbol{x}$, onde $\boldsymbol{w^{(c)}}$ é a matriz dos pesos sinápticos que se ligam à camada *c*;
2. Aplica-se a função de ativação na saída da camada *c*: $\boldsymbol{y^{(c)}} = \boldsymbol{\varphi(v_{ck})}$;

3. Para ocorrer a propagação, a saída da camada *c* tem que ser a entrada da camada *c+1*: $\boldsymbol{v_{(c+1)k} = w^{(c+1)^T} . y^{(c)}}$;
4. Aplica-se a função de ativação na saída da camada l: $\boldsymbol{y^{(c+1)} = \varphi(v_{(c+1)k})}$;
5. Repete-se esse procedimento até chegar na camada de saída;

Pode-se usar operações e estruturas matriciais para facilitar os cálculos computacionais do sistema, e dessa forma, colocar todas as entradas em uma matriz de entradas, ao invés de tratar um vetor por vez, facilita os cálculos da propagação. Da mesma maneira, os pesos podem ser colocados em forma matricial para representar todas as conexões sinápticas de uma camada neuronal a outra.

Considerando que se têm *m* amostras de sinais de entrada e cada sinal de entrada tem *n* parâmetros, pode-se representar essas *m* amostras como se segue:

$$\boldsymbol{x} = [\boldsymbol{x^{(1)}} \quad \boldsymbol{x^{(2)}} \quad \boldsymbol{x^{(3)}} \quad \dots \quad \boldsymbol{x^{(m)}}] = \begin{bmatrix} x_0^{(1)} & x_0^{(2)} & \dots & x_0^{(m)} \\ x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(m)} \\ x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(m)} \\ \vdots & \vdots & \cdots & \vdots \\ x_n^{(1)} & x_n^{(2)} & \dots & x_n^{(m)} \end{bmatrix} \qquad (25)$$

Pode-se observar que os vetores $\boldsymbol{x^{(m)}}$ são vetores coluna, como mostrado na Seção 3.1., em que, *m* é o número de sinais, e *n* o número de parâmetros no sinal. Os pesos $\boldsymbol{w_k}^{(c)}$ podem ser representados na forma de vetores coluna, em uma arrumação matricial, onde *k* é o neurônio onde os respectivos pesos do vetor estão conectados, como é mostrado na Eq. (26):

$$\boldsymbol{w^{(c)}} = \left[\boldsymbol{w_1}^{(c)} \quad \boldsymbol{w_2}^{(c)} \quad \boldsymbol{w_3}^{(c)} \quad \dots \quad \boldsymbol{w_k}^{(c)}\right] = \begin{bmatrix} w_{10}^{(c)} & w_{20}^{(c)} & \dots & w_{k0}^{(c)} \\ w_{11}^{(c)} & w_{21}^{(c)} & \dots & w_{k1}^{(c)} \\ w_{12}^{(c)} & w_{22}^{(c)} & \dots & w_{k2}^{(c)} \\ \vdots & \vdots & \cdots & \vdots \\ w_{1n}^{(c)} & w_{2n}^{(c)} & \dots & w_{kn}^{(c)} \end{bmatrix} \qquad (26)$$

Portanto, para encontrar o $\boldsymbol{v_{ck}}$ (campo local induzido) de todos os neurônios da camada, para todos os sinais, faz-se o produto:

$$v_{ck} = \boldsymbol{w}^{(c)^T}.\boldsymbol{x} = \begin{bmatrix} \boldsymbol{w_1}^{(c)^T}\boldsymbol{x}^{(1)} & \boldsymbol{w_1}^{(c)^T}\boldsymbol{x}^{(2)} & \dots & \boldsymbol{w_1}^{(c)^T}\boldsymbol{x}^{(m)} \\ \boldsymbol{w_2}^{(c)^T}\boldsymbol{x}^{(1)} & \boldsymbol{w_2}^{(c)^T}\boldsymbol{x}^{(2)} & \dots & \boldsymbol{w_2}^{(c)^T}\boldsymbol{x}^{(m)} \\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{w_k}^{(c)^T}\boldsymbol{x}^{(1)} & \boldsymbol{w_k}^{(c)^T}\boldsymbol{x}^{(2)} & \dots & \boldsymbol{w_k}^{(c)^T}\boldsymbol{x}^{(m)} \end{bmatrix} = \begin{bmatrix} v_{c1}^{(1)} & v_{c1}^{(2)} & \dots & v_{c1}^{(m)} \\ v_{c2}^{(1)} & v_{c2}^{(2)} & \dots & v_{c2}^{(m)} \\ \vdots & \vdots & \vdots & \vdots \\ v_{ck}^{(1)} & v_{ck}^{(2)} & \dots & v_{ck}^{(m)} \end{bmatrix} \quad (27)$$

Dessa forma, pode-se computar o campo local induzido de todas as amostras de dados a serem treinados de uma única vez e aplicar o algoritmo previamente definido para propagar os sinais de entrada até a camada de saída. A Figura 10 mostra a representação gráfica de uma rede neural multicamadas.

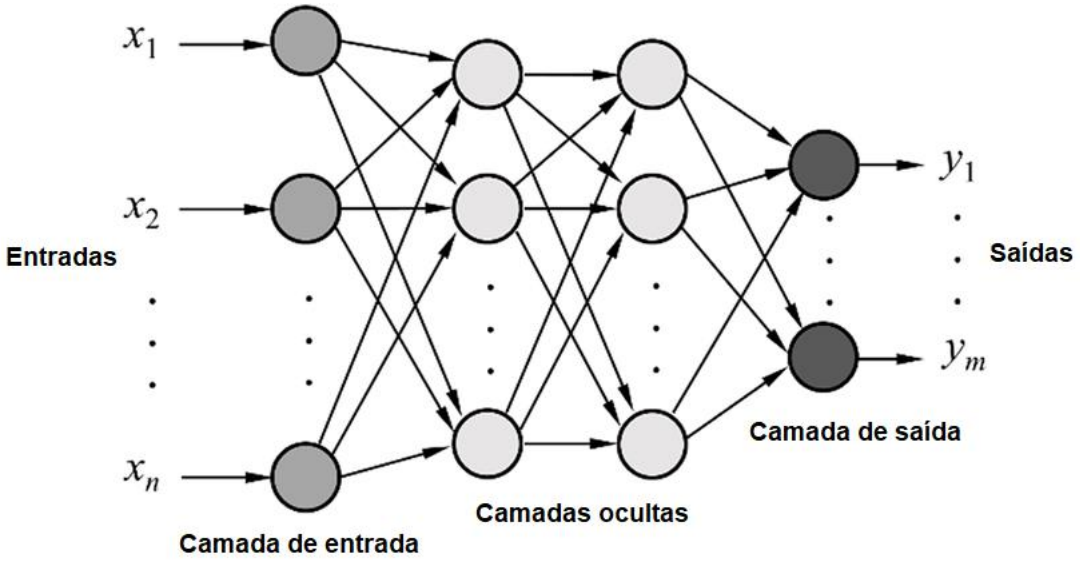

Figura 10 – Rede Neural multicamadas. Fonte: Pouliakis (2016).

3.4. Aprendizado de uma rede neural

Uma das propriedades de maior importância em uma rede neural é a sua capacidade de aprender a partir do seu ambiente e, melhorar seu desempenho com o aprendizado (HAYKIN, 2001). O aprendizado de uma rede neural se dá a partir do ajuste de pesos e bias feito por um processo de otimização iterativo.

A aprendizagem no contexto das redes neurais segundo Mendel e McClarem (1970), corresponde ao: "*processo pelo qual os parâmetros livres de uma rede neural são adaptados através de um processo de estimulação pelo ambiente no qual a rede*

está inserida. O tipo de aprendizagem é determinado pela maneira pela qual a modificação dos parâmetros ocorre" (HAYKIN, 2001).

Dessa forma, pode-se dizer que o processo de aprendizado é feito da seguinte forma (HAYKIN, 2001):

1. A rede neural é estimulada por um ambiente;
2. A partir desse estimulo a rede sofre modificação em seus parâmetros livres;
3. A rede passa a responder de forma diferente ao ambiente devido às alterações na estrutura de seus parâmetros livres.

A partir dessas definições e tendo conhecimento da alteração dos parâmetros livres da rede para que o aprendizado ocorra, é definida uma medida de eficácia dessas alterações. Essa medida é denominada de função de custo, em que o objetivo de um algoritmo de aprendizagem é a redução a cada iteração dessa função de custo.

Segundo Simon Haykin, em seu livro "*redes neurais: Princípios e Práticas*", três dos principais tipos de processos de aprendizado de uma rede neural são (HAYKIN, 2001):

- Aprendizado supervisionado: é apresentado um conjunto de treino, consistindo de entradas e saídas desejadas;
- Aprendizado por reforço: para cada entrada apresentada, é produzida uma indicação (reforço) sobre a adequação das saídas correspondentes produzidas pela rede (interação contínua com o ambiente);
- Aprendizado não-supervisionado: a rede atualiza seus pesos sem o uso de pares de entrada-saídas desejadas e sem indicações sobre a adEq. das saídas produzidas, ao invés disso, são dadas condições da rede realizar uma medida da representação que a rede deve aprender, e seus parâmetros serão atualizados em relação a essa medida.

3.4.1. Função de custo Média dos Quadrados

A função de custo Média dos Quadrados, chamada também de Erro Quadrático Médio (RAUBER, 2015), tem por objetivo medir a distância entre o valor real de uma

saída dada as entradas, e o valor que a rede neural estabeleceu como saída ao receber essas mesmas entradas. Quanto menor for o Erro Quadrático Médio, menor será a distância entre o valor real e o valor alcançado pela rede neural em questão.

Dessa forma, podemos definir o Erro Quadrático Médio como (RAUBER, 2015):

$$C(w,b) = \frac{1}{2n}\sum_{j=1}^{n} \left\| y_j^{(o)} - t_j \right\|^2 \qquad (28)$$

onde $y_j^{(o)}$ é a saída da camada, o resultado da amostra de dados de entrada j e t_j é o valor alvo de saída para a amostra de entrada j e C é uma função dependente de todos os pesos e bias.

3.4.2. Função de custo Entropia Cruzada

A função de custo Entropia Cruzada (TICHONOV, 2017), tem como objetivo medir a eficácia de um modelo de classificação usado por uma rede neural, aonde as saídas da rede neural são probabilidades. Dessa forma, a função de custo é reduzida conforme a rede neural tem saídas mais próximas da classe a que deveria pertencer.

Dessa forma, pode-se definir a função de Entropia Cruzada como (TICHONOV, 2017):

$$C = -\frac{1}{m}\sum_{x=1}^{m}\sum_{j=1}^{n}\left[{t_j}^x \ln\left(y_j^{x(o)}\right) + \left(1 - {t_j}^x\right)\ln\left(1 - y_j^{x(o)}\right)\right] \qquad (29)$$

3.4.3. Gradiente descendente

Quando se busca minimizar uma função de custo convexa (encontrar o mínimo da função), pode-se começar em um ponto arbitrário e movimentar-se ao longo do gradiente até o próximo ponto e repetir isso até um ponto estacionário, ou seja, onde o gradiente é zero (SINGER, 2016).

Pode-se observar este movimento ao longo do gradiente na Figura 11.

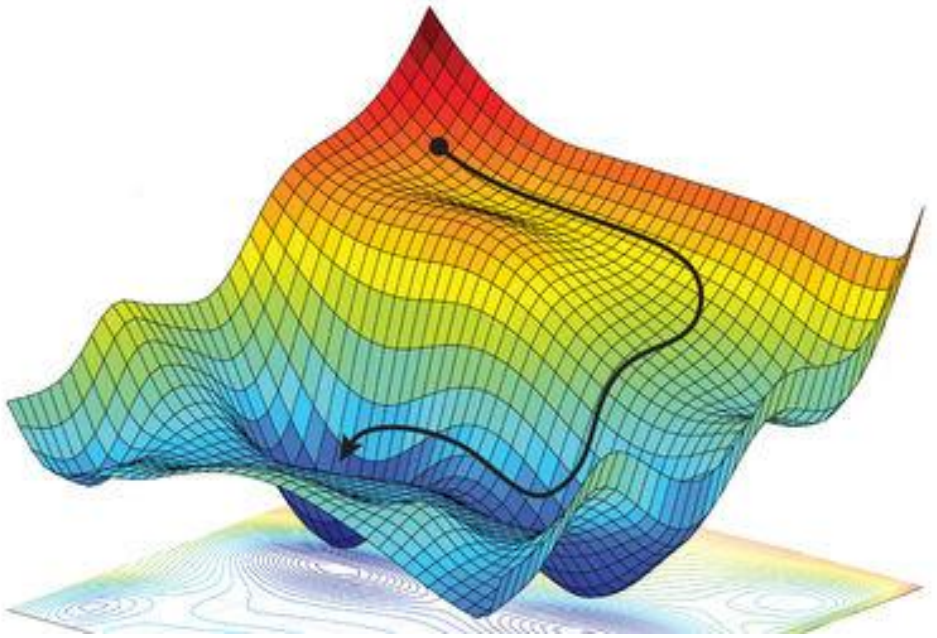

Figura 11 – Demonstração visual do movimento ao longo do gradiente. Fonte: Amini (s/d).

Assim, pode-se dizer que o modelo de otimização dos parâmetros livres de uma rede neural, conhecido como Gradiente Descendente, atualiza os pesos e os bias da rede de forma a minimizar a função de custo; para isso, se utiliza do gradiente da função:

$$w'_{kj} \rightarrow w_{kj} - \eta \nabla C(w,b) \tag{30}$$

onde η é um parâmetro ajustável chamado de taxa de aprendizagem, que representa o tamanho dos passos que serão dados ao longo e no sentido contrário do gradiente a cada iteração. Dessa forma, pode-se definir o gradiente da função de custo para uma determinada camada m como:

$$\nabla C(w,b)^{(m)} = \frac{\partial C(w,b)}{\partial {w_{kj}}^{(m)}} = \frac{\partial C(w,b)}{\partial {y_k}^{(m)}} \frac{\partial {y_k}^{(m)}}{\partial {v_k}^{(m)}} \frac{\partial {v_k}^{(m)}}{\partial {w_{kj}}^{(m)}} \tag{31}$$

em que para a camada de saída o, tem-se:

$$\frac{\partial {v_k}^{(o)}}{\partial {w_{kj}}^{(o)}} = {y_k}^{(o-1)} \tag{32}$$

$$\frac{\partial {y_k}^{(o)}}{\partial {v_k}^{(o)}} = \varphi'(v_k^{(o)}) \tag{33}$$

O termo $\frac{\partial C(w,b)}{\partial {y_k}^{(m)}}$ da Eq. (31) é obtido através da derivação simples da função de custo escolhida. E para uma camada intermediaria *i* tem-se que:

$$\frac{\partial {v_k}^{(i)}}{\partial {w_{kj}}^{(i)}} = {y_k}^{(i-1)} \tag{34}$$

$$\frac{\partial {y_k}^{(i)}}{\partial {v_k}^{(i)}} = \varphi'(v_k^{(i)}) \tag{35}$$

$$\frac{\partial C(w,b)}{\partial {y_k}^{(i)}} = \sum_{j=1}^{n} \frac{\partial C(w,b)}{\partial v_{kj}^{(o)}} \frac{\partial v_{kj}^{(o)}}{\partial y_k} \tag{36}$$

onde, *j* representa as amostras de dados de entrada.

3.4.4. Otimizador ADAM (*Adaptative Moment Estimation*)

O otimizador ADAM foi proposto por Diederik Kingma e Jimmy Ba em 2014, como alternativa ao otimizador de Gradiente Descendente padrão, uma vez que este mantém constante durante todo o treinamento a taxa de aprendizagem, já o ADAM atualiza a taxa de aprendizagem a cada iteração (KINGMA e BA, 2015).

O otimizador ADAM pode ser definido matematicamente segundo as Eqs. (37), (38) e (39) (KESKAR e SOCHER, 2017):

$$w_k = w_{k-1} - \alpha_{k-1} \frac{\sqrt{1-\beta_2^k}}{1-\beta_1^k} \frac{m_{k-1}}{\sqrt{v_{k-1}} + \epsilon} \tag{37}$$

onde,

$$m_{k-1} = \beta_1 w_{k-2} + (1-\beta_1)\widehat{\nabla} f(w_{k-1}) \tag{38}$$

$$v_{k-1} = \beta_2 v_{k-2} + (1-\beta_2)\widehat{\nabla} f(w_{k-1})^2 \tag{39}$$

β_1 e β_2 são as taxas de decaimento exponencial para as estimativas de momento, f é a função a ser minimizada, α é a taxa de aprendizagem, m_k e v_k são os momentos e ϵ é um parâmetro para evitar um erro por divisão por zero.

3.5. TENSORFLOW

O TensorFlow é uma biblioteca lançada pela equipe *Google Brain*, do *Google*, e é usada internamente na empresa tanto para pesquisa quanto para produção, em que o programador define as operações matemáticas que devem ser realizadas por meio de grafos, que são levados a um código otimizado e compilado em C++ para realizar a computação das operações antes definidas (GÉRON, 2017). Seu objetivo é facilitar o desenvolvimento de redes neurais profundas (ABADI, *et al.*, 2016) (número de camadas ocultas superior a 1) que usam grande quantidade de dados, como sistemas capazes de detectar objetos em imagens, sistemas que entendem a fala humana, análise de vídeo, propriedades de remédios potenciais, entre outras tantas aplicações que demandam tais características. O TensorFlow é versátil e pode ser usado de forma nativa em diversas plataformas, com suporte em *datacenters, clusters, desktops,* e em dispositivos móveis em geral (ABADI *et al.*, 2016). Uma grande vantagem do TensorFlow é o fato de poder ter seus cálculos realizados na GPU, que os executa de forma paralela, fazendo com que o poder de processamento aumente significativamente.

Com a possibilidade da utilização de uma linguagem de alto nível no desenvolvimento de aplicações, a biblioteca ganha destaque ao poder ser utilizada em diversos modelos de aprendizagem e treinamento sem interferir no funcionamento padrão dos sistemas em que esta sendo utilizada.

Em Géron (2017), pode-se observar um exemplo da forma visual de uma computação matemática na forma de grafo no TensorFlow, Fig.12.

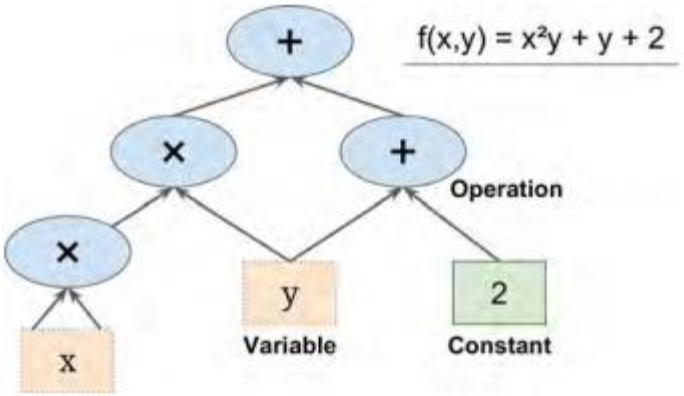

Figura 12 – Exemplo de grafo de computação. Fonte: Géron (2017).

A biblioteca TensorFlow oferece os seguintes benefícios (GÉRON, 2017; ABADI *et al.*, 2016):

1. Grande flexibilidade na criação de diversas topologias de redes neurais;
2. Alta eficiência devido à computação dos grafos na linguagem de programação C++;
3. Calculo automático dos gradientes das funções que o programador define, além de métodos avançados de otimização dos parâmetros que minimizam a função de custo.
4. Múltiplas execuções simultâneas devido aos seus subgrafos;
5. Grande variedade de funções de ativação;
6. Grande variedade de métodos de aprendizado;

CAPÍTULO 4
METODOLOGIA

A metodologia utilizada na elaboração desse trabalho se desenvolveu em quatro etapas: i) uso da evolução temporal das variáveis de estado de uma central nuclear para três casos de acidente e o caso de operação normal simulados por Alvarenga, 1997; ii) implementação das redes neurais utilizando o framework Tensorflow; iii) implementação de um programa para fazer testes automatizados com diversas topologias de redes neurais e iv) implementação de um programa para seleção das melhores combinações de variáveis de estado. Essas etapas são apresentadas de forma resumida na Figura 13:

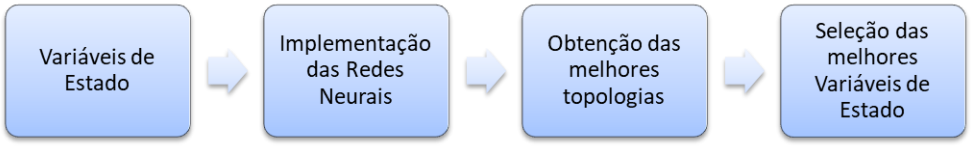

Figura 13 – Fluxograma das etapas do trabalho.

4.1 Variáveis de estado

Variáveis de estado são um conjunto de variáveis que determinam matematicamente o estado de um sistema dinâmico. Dessa forma, essas variáveis permitem conhecer o estado de um sistema de forma a conseguir determinar o seu comportamento futuro (PALM, 2010).

Neste trabalho, foram usadas 17 variáveis de estado mais a variável tempo simuladas para a central nuclear de Angra 2, que em principio são julgados como suficientes para o entendimento do evento em curso. A evolução temporal de cada variável foi obtida por Alvarenga (1997), em linguagem MATLAB 4.0, e o tempo total de evolução de cada variável foi de 61 segundos onde o primeiro segundo corresponde à condição normal de potência do reator e o segundo seguinte corresponde ao TRIP do reator. O tempo de 61 segundos foi utilizado, pois este tempo é o suficiente para haver uma diferenciação entre os cenários estudados, porém, no presente trabalho, o primeiro segundo dos cenários foi eliminado por corresponder ao TRIP da planta e ser igual para as quatro situações estudadas, desta forma gerando

confusão desnecessária no treinamento das redes neurais que se seguiriam a partir destes dados. As variáveis simuladas são mostradas na Tabela 2:

Tabela 2 – Variáveis Simuladas.

1	Vazão do núcleo (%)
2	Temperatura da perna quente (°C)
3	Temperatura da perna fria (°C)
4	Vazão no núcleo (kg/s)
5	Nível do 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 de ruptura (kg/s)
11	Vazão no circuito primário (kg/s)
12	Tempo (s)
13	Pressão no sistema primário (MPa)
14	Potência térmica (%)
15	Potência nuclear (%)
16	Margem de sub-resfriamento (°C)
17	Nível do pressurizador (%)
18	Temperatura média no primário (°C)

Fonte: Nicolau (2014).

4.2. Implementação das Redes Neurais

Neste trabalho, foi usada a linguagem de programação Python juntamente com uma biblioteca de código aberto focada na computação numérica e aprendizado de máquina, chamada TensorFlow (GÉRON, 2017).

A construção da metodologia proposta usando rede neural em Python e a biblioteca TensorFlow se deu seguindo os seguintes passos:

1. Importação para o ambiente de desenvolvimento das variáveis de estado;
2. Criação do arquivo para armazenar os resultados da rede neural;

3. Seleção das variáveis de estado que seriam utilizadas na computação da rede neural;
4. Normalização das variáveis de estado utilizadas na computação;
5. Separação dos dados em dados de treinamento e dados de teste;
6. Criação de *placeholders* para armazenar o resultado das operações e os valores de entrada de cada rodada de treinamento;
7. Criação das variáveis Peso e *Bias* de cada uma das camadas; nesse estudo foram utilizadas zero, uma e duas camadas ocultas;
8. Definição do grafo das operações matemáticas como foram apresentadas nas Eqs. (3) e (4), que será usado para computar a alimentação de avanço da rede neural;
9. Definição da função de custo que deverá ser minimizada durante o treinamento da rede neural, a função de custo utilizada foi a função de entropia cruzada;
10. Definição do processo de aprendizagem que será utilizado na otimização dos parâmetros livres da rede neural (Pesos e Bias), em que, o Otimizador ADAM foi utilizado com taxa de aprendizagem de 0,01 (valor padrão utilizado na função do otimizador);
11. Inicialização da seção no TensorFlow;
12. Definição do número de épocas usadas no treinamento, onde foram utilizadas 10000 épocas para treinar a rede neural;
13. Iniciação do processo de treinamento;
14. Utilização dos dados de teste para verificar a precisão da rede neural já treinada.
15. Registro dos parâmetros e da precisão da Rede Neural no arquivo criado para armazenamento de resultados.

4.3. Teste de Topologias e variáveis de estado

Para se obter o melhor desempenho na classificação dos cenários de uma central nuclear, a topologia da rede neural deve ser escolhida de forma a maximizar a sua precisão. Para isso, foram desenvolvidos dois programas em Python para automatizar o processo de teste de topologias e posteriormente para encontrar as combinações mais adequadas de variáveis de estado para cada topologia selecionada.

O primeiro programa foi usado para identificar as topologias para os conjuntos de 4, 6, 8 e 10 entradas da rede, com zero, uma e duas camadas ocultas com diferentes funções de ativação (Softmax e Sigmoide), esse programa encerra a execução ao atingir o número máximo de neurônios. A Figura 14 ilustra o funcionamento desse programa.

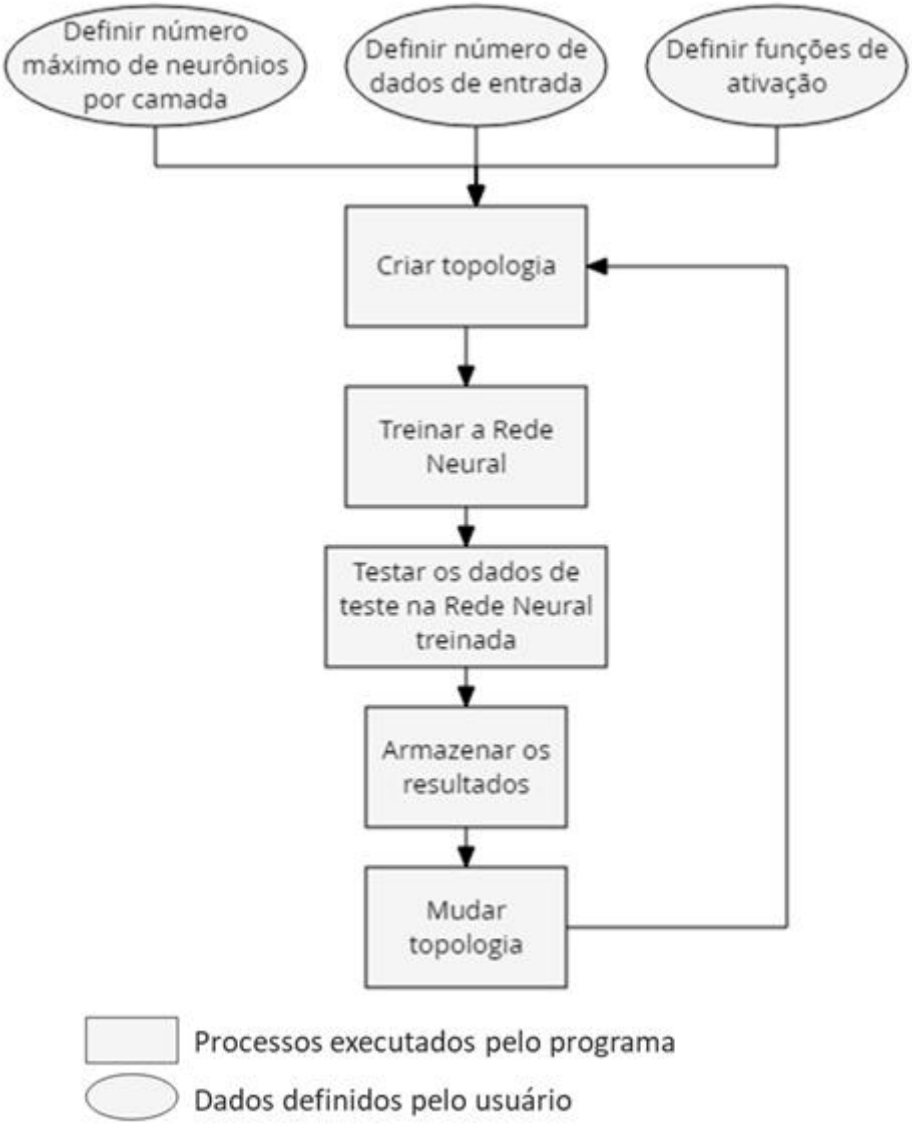

Figura 14 – Processo de execução do programa de teste de topologias.

O segundo programa foi utilizado para encontrar a combinação mais adequada de variáveis de estado para cada topologia selecionada para encontrar o menor conjunto de variáveis de estado que melhor otimizasse ou mantivesse em níveis de

precisão elevados (diferenciando assim os estados da planta com o menor número de informações) as topologias selecionadas no teste 1. Como testar todas as possíveis combinações de variáveis de estado para redes com número de neurônios na camada de entrada igual a 4, 6, 8 e 10 seria computacionalmente inviável, foram escolhidas aleatoriamente 100 conjuntos de variáveis para cada topologia selecionada no teste 1 de maneira a representar da melhor forma possível o total de combinações existentes. O programa se encerra quando atinge o número de combinações definidas. A Figura 15 representa o funcionamento do programa.

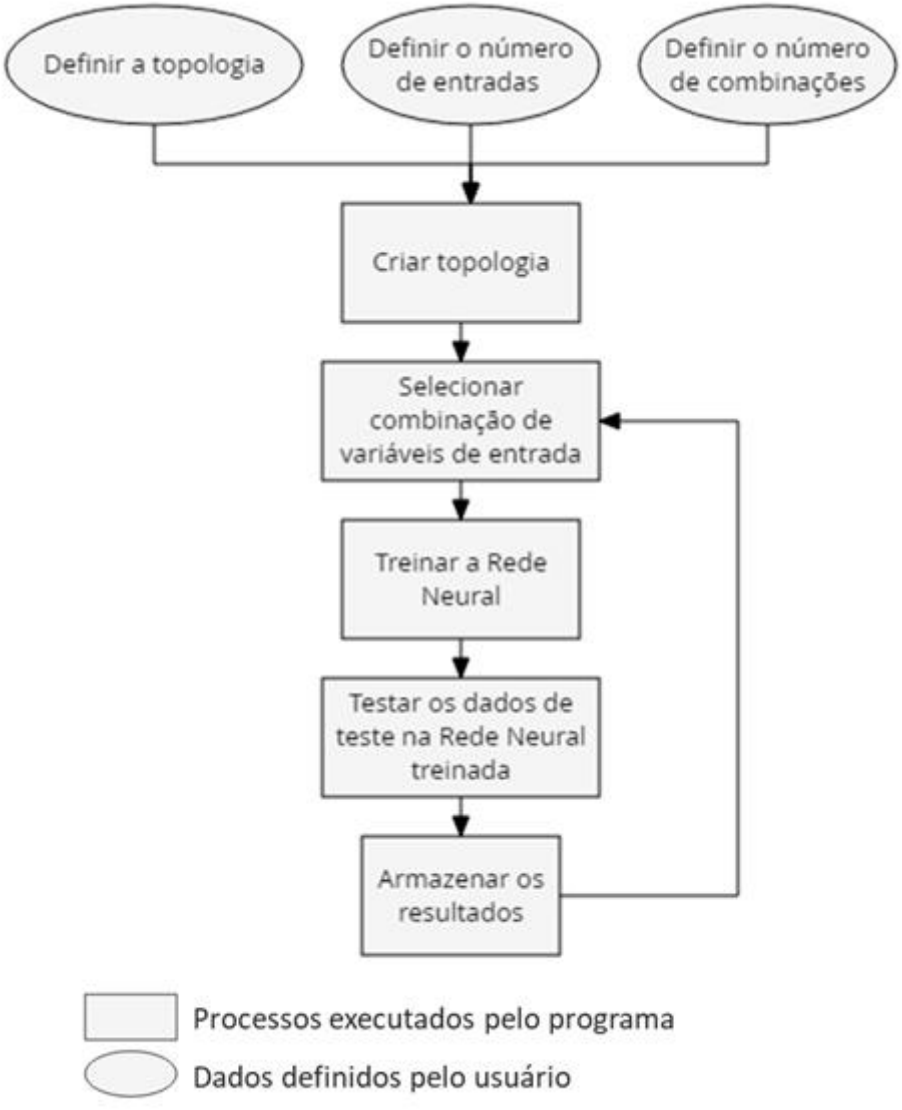

Figura 15 – Processo execução do programa de teste de combinações de variáveis de entrada.

CAPÍTULO 5
RESULTADOS E DISCUSSÕES

Com a elaboração do programa de teste de topologias, foram obtidas as topologias com maior precisão, ou seja, as que depois de treinadas acertaram maior número de cenários do conjunto de teste (validação). Foram selecionadas no Teste 1 topologias para os conjuntos de 4, 6, 8 e 10 variáveis de entrada e diferentes combinações de funções de ativação. Como algumas topologias obtiveram precisão igual, optou-se no presente trabalho por mostrar apenas as 6 com maior eficiência, ou seja, as que apresentaram o mesmo resultado com o menor número total de neurônios ou variáveis de estado.

Portanto, podem-se observar na Tabela 3 as topologias que obtiveram as melhores precisões de classificação para uma rede neural de duas camadas ocultas, em que todos os neurônios da primeira e segunda camada usam a função sigmoide como função de ativação e os neurônios da camada de saída usam a função *Softmax*. É possível observar que as melhores topologias neuronais e suas respectivas precisões ocorrem para números acima de oito variáveis de entrada, mostrando que com essa escolha de funções de ativação, a rede necessita de mais variáveis para identificar de forma mais precisa a separação entre classes, porém, a mesma não se compromete em precisão ao usar menos variáveis de entrada se a topologia for escolhida de forma correta, o que mostra o resultado das primeira, segunda, terceira e quarta linhas da Tabela 3, que apesar de menos precisos, atingem precisões significativas.

Tabela 3 – Topologias (Sigmoide-Sigmoide-*Softmax*).

Nº de variáveis	***Nº de Neurônios Camada 1***	***Nº de Neurônios Camada 2***	***Precisão***
4	1	1	92,50%
4	2	1	92,50%
4	3	1	92,50%
4	10	2	92,50%
6	1	2	92,50%
6	10	2	92,50%
8	1	3	96,25%
8	1	6	96,25%
8	2	3	96,25%
8	2	5	96,25%
8	2	6	96,25%
8	4	3	96,25%
10	1	2	96,25%
10	1	3	96,25%
10	1	6	96,25%
10	2	2	96,25%
10	2	3	96,25%
10	2	6	96,25%

Na Tabela 4 são apresentadas as topologias que obtiveram melhores precisões de classificação para uma rede neural de duas camadas ocultas, em que todos os neurônios da primeira, segunda, e terceira camadas usam a função *Softmax* como função de ativação. Nesta rede pode-se observar a maior precisão encontrada na topologia que se utilizou do menor número de variáveis de entrada, e a média das precisões aumentando em relação à rede neural utilizada para a construção da Tabela 3, mudança essa vinda em decorrência da troca de funções de ativação, onde a função Softmax utilizada em todos os neurônios da rede se demonstrou mais eficaz.

Tabela 4 – Topologias (*Softmax- Softmax- Softmax*).

Nº de variáveis	***Nº de Neurônios Camada 1***	***Nº de Neurônios Camada 2***	***Precisão***
4	7	5	97,50%
6	5	9	96,25%
8	2	9	96,25%
8	2	11	96,25%
8	3	5	96,25%
8	5	9	96,25%
8	5	11	96,25%
8	6	9	96,25%
10	2	5	96,25%
10	2	7	96,25%
10	2	9	96,25%
10	2	10	96,25%
10	5	6	96,25%
10	5	7	96,25%

Na Tabela 5 são apresentadas as topologias que obtiveram melhores precisões de classificação para uma Rede Neural de duas camadas ocultas, em que todos os neurônios da primeira camada usam a função sigmoide como função de ativação e os neurônios da segunda camada e camada de saída usam a função *Softmax* como função de ativação. A rede neural cujos resultados estão expressos na Tabela 5, têm um comportamento muito próximo dos da Tabela 3, em que redes com número maior ou igual a oito variáveis de entrada tem precisão maior. A escolha da função sigmoide contribui para essa perda de precisão quando o número de variáveis de entrada é menor.

Tabela 5 –Topologias (Sigmoide-Softmax-Softmax).

Nº de variáveis	***Nº de Neurônios Camada 1***	***Nº de Neurônios Camada 2***	***Precisão***
4	7	6	92,50%
4	8	6	92,50%
4	9	6	92,50%
4	9	7	92,50%
4	10	5	92,50%
6	4	6	92,50%
6	8	5	92,50%
6	8	6	92,50%
6	9	5	92,50%
8	1	7	96,25%
8	2	5	96,25%
8	2	7	96,25%
8	3	5	96,25%
8	3	7	96,25%
8	4	5	96,25%
10	1	3	96,25%
10	1	6	96,25%
10	1	7	96,25%
10	2	5	96,25%
10	2	6	96,25%
10	4	7	96,25%

Na Tabela 6 são apresentadas as topologias que obtiveram melhores precisões de classificação para uma Rede Neural de duas camadas ocultas, em que todos os neurônios da primeira camada e camada de saída usam a função *Softmax* como função de ativação e os neurônios da segunda camada usam a função Sigmoide como função de ativação. A rede neural expressa na Tabela 6 apresenta valores de precisão máxima para valores de variáveis de entrada maiores ou iguais 6 nas topologias adequadas, o que nos faz concluir que a função Softmax mais próxima da camada de entrada aumenta a precisão para redes classificadoras com combinações de funções.

Tabela 6 – Topologias (Softmax-Sigmoide- Softmax).

Nº de variáveis	***Nº de Neurônios Camada 1***	***Nº de Neurônios Camada 2***	***Precisão***
4	10	9	95,00%
6	2	9	96,25%
6	3	5	96,25%
6	7	9	96,25%
8	2	7	96,25%
8	2	10	96,25%
8	3	3	96,25%
8	3	6	96,25%
8	3	7	96,25%
8	3	10	96,25%
10	2	5	96,25%
10	2	6	96,25%
10	2	7	96,25%
10	3	3	96,25%
10	3	6	96,25%
10	3	7	96,25%

Na Tabela 7 são apresentadas as topologias que obtiveram melhores precisões de classificação para uma rede neural de uma camada oculta, em que todos os neurônios da primeira camada e camada de saída usam a função *Softmax* como função de ativação. A Tabela 7 nos mostra que a combinação de duas funções Softmax em topologias de redes neurais de uma camada traz resultados acima de 90% de precisão apesar da redução de uma camada oculta em relação aos resultados dos testes anteriormente apresentados.

Tabela 7 – Topologias (Softmax- Softmax).

Nº de variáveis	***Nº de Neurônios Camada 1***	***Precisão***
4	7	95,00%
6	7	93,75%
6	9	93,75%
6	10	93,75%
8	3	93,75%
8	6	93,75%
8	9	93,75%
8	10	93,75%
10	3	92,50%
10	5	92,50%
10	6	92,50%
10	7	92,50%
10	9	92,50%
10	11	92,50%

Na Tabela 8 são apresentadas as topologias que obtiveram melhores precisões de classificação para uma rede neural de uma camada oculta, em que todos os neurônios da primeira usam a função sigmoide como função de ativação e os neurônios da camada de saída usam a função *Softmax* como função de ativação. Esta rede neural com apenas uma camada oculta, mostra que o maior valor obtido de precisão se deu para 8 variáveis e um neurônio na camada oculta, o que nos faz pensar que nessa combinação de funções de ativação e nas combinações de variáveis escolhidas para achar a melhor topologia neuronal, a função sigmoide contribui de forma sensível para o aumento de precisão da rede.

Tabela 8 – Topologias (Sigmoide - Softmax).

Nº de variáveis	***Nº de Neurônios Camada 1***	***Precisão***
4	9	93,75%
6	10	95,00%
8	1	96,25%
10	5	93,75%
10	6	93,75%
10	9	93,75%
10	12	93,75%

Na Tabela 9 são apresentadas as precisões obtidas quando não se utiliza nenhuma camada oculta na rede neural, em que os neurônios da camada de saída usam a função *Softmax* como função de ativação. Nesta configuração de rede é tirada a conclusão mais importante, pois a rede consegue obter considerável precisão sem necessitar para isso de nenhuma camada oculta, oque em outras palavras, quer dizer que os dados utilizados para testar a topologia são linearmente separáveis, e portanto, existem variáveis de estado que conseguem separar de forma completa todos os cenários da central nuclear para os casos estudados.

Tabela 9 – Precisão de uma Rede Neural sem camadas ocultas (*Softmax*).

Nº de variáveis	***Precisão***
4	93,75%
6	88,75%
8	86,25%
10	85,00%

Utilizando o programa para teste das possíveis combinações de variáveis de entrada que otimizassem o resultado da rede neural, foi necessário escolher uma amostragem de 100 combinações de variáveis para os casos de 4, 6, 8 e 10 devido ao alto número de combinações possíveis.

Portanto, na Tabela 10 são apresentadas algumas das variáveis que no mínimo mantêm a precisão da rede neural de duas camadas cujas primeira e segunda camada são compostas por neurônios sigmoidais e, em sua camada de saída, neurônios que utilizam a função de ativação *Softmax.*

Tabela 10 – variáveis otimizadoras da Rede Neural (Sigmoide – Sigmoide – *Softmax*).

Nº de entradas	***variáveis***	***Precisão***
4	6, 9, 10 e 16	95,00%
6	1, 2, 4, 10, 13 e 15	96,25%
6	2, 8, 10, 11, 12 e 18	96,25%
6	1, 9, 10, 14, 17 e 18	96,25%
6	2, 5, 6, 9, 10 e 12	96,25%
6	1, 6, 10, 11, 14 e 16	96,25%
6	2, 10, 11, 12, 14 e 18	96,25%
6	4, 5, 6, 10, 12 e 14	96,25%
6	1, 2, 10, 12, 14 e 18	96,25%
8	6, 8, 10, 11, 12, 13, 17 e 18	96,25%
8	1, 4, 5, 8, 9, 10, 15 e 17	96,25%
8	5, 10, 11, 12, 13, 15, 16 e 17	96,25%
8	1, 4, 10, 11, 14, 15, 17 e 18	96,25%
8	2, 6, 9, 10, 13, 15, 16 e 18	96,25%
8	4, 9, 10, 12, 13, 14, 16 e 18	96,25%
8	1, 6, 10, 13, 14, 15, 17 e 18	96,25%
8	1, 6, 8, 10, 11, 13, 14 e 16	96,25%
8	1, 6, 8, 10, 11, 14, 16 e 17	96,25%
8	4, 7, 9, 10, 14, 15, 16 e 17	96,25%
10	1, 4, 6, 8, 10, 12, 13, 14, 15 e 16	96,25%
10	1, 2, 7, 8, 10, 13, 14, 15, 16 e 17	96,25%
10	1, 2, 4, 8, 10, 11, 12, 13, 14 e 17	96,25%
10	6, 7, 8, 9, 10, 11, 13, 14, 16 e 17	96,25%
10	1, 2, 3, 4, 8, 9, 10, 12, 13 e 18	96,25%

Na Tabela 11, são apresentadas algumas das variáveis que, no mínimo, mantêm a precisão da rede neural de duas camadas ocultas cujas primeira, segunda e a camada de saída são compostas por neurônios que utilizam a função de ativação *Softmax*. No teste de combinação de variáveis desta rede, podemos ver que os valores de precisão da Tabela 4 já estavam saturados, mantendo-se inalterados.

Tabela 11 – variáveis otimizadoras da Rede Neural (*Softmax* – *Softmax* – *Softmax*).

.Nº de entradas	***variáveis***	***Precisão***
4	4, 7, 10 e 17	97,50%
6	2, 5, 10, 13, 14, e 15	96,25%
6	6, 9, 10, 12, 13, e 14	96,25%
6	2, 10, 14, 15, 17, e 18	96,25%
6	2, 8, 10, 14, 15, e 18	96,25%
6	8, 9, 10, 11, 12, e 13	96,25%
8	1, 2, 5, 8, 10, 11, 12, e 18	96,25%
8	2, 3, 4, 8, 9, 10, 11, e 15	96,25%
8	2, 4, 8, 9, 10, 11, 12, e 16	96,25%
8	1, 2, 4, 10, 12, 14, 16, e 18	96,25%
8	1, 2, 4, 5, 8, 9, 10, e 12	96,25%
8	2, 6, 10, 11, 13, 14, 15, e 17	96,25%
8	5, 8, 9, 10, 11, 12, 14, e 16	96,25%
8	4, 5, 8, 10, 12, 15, 17, e 18	96,25%
8	1, 2, 8, 9, 10, 12, 14, e 16	96,25%
8	1, 6, 10, 11, 12, 14, 16, e 18	96,25%
10	3, 4, 10, 12, 13, 14, 15, 16, 17, e 18	96,25%
10	1, 3, 4, 5, 6, 8, 10, 12, 13, e 17	96,25%
10	1, 2, 4, 6, 8, 10, 11, 14, 16, e 17	96,25%
10	3, 5, 6, 10, 12, 14, 15, 16, 17, e 18	96,25%
10	1, 3, 4, 5, 7, 9, 10, 11, 13, e 16	96,25%
10	2, 3, 4, 8, 10, 12, 14, 16, 17, e 18	96,25%
10	2, 9, 10, 11, 12, 13, 14, 15, 16, e 17	96,25%
10	4, 5, 7, 9, 10, 12, 14, 15, 16, e 17	96,25%
10	1, 4, 5, 6, 8, 9, 10, 12, 13, e 15	96,25%
10	1, 2, 6, 9, 10, 12, 14, 15, 16, e 17	96,25%
10	2, 3, 4, 5, 8, 10, 11, 13, 15, e 16	96,25%

Na Tabela 12 são apresentadas algumas das variáveis que, no mínimo, mantêm a precisão da rede neural de duas camadas, em que a primeira camada é composta por neurônios sigmoidais e a segunda é composta por neurônios que utilizam a função de ativação *Softmax*. Pode-se observar que no teste desta rede, as topologias de quatro e seis variáveis de entrada foram otimizadas, enquanto as outras se mantiveram inalteradas se comparadas as da Tabela 5.

Tabela 12 – variáveis otimizadoras da Rede Neural (Sigmoide – *Softmax* – *Softmax*).

Nº de entradas	***variáveis***	***Precisão***
4	6, 8, 9, e 10	96,25%
4	9, 10, 11, e 14	96,25%
4	1, 9, 10, e 18	96,25%
6	1, 2, 10, 11, 15, e 16	96,25%
6	2, 6, 10, 11, 12, e 14	96,25%
8	4, 6, 8, 9, 10, 13, 14, e 16	96,25%
8	4, 6, 8, 9, 10, 11, 14, e 16	96,25%
8	4, 6, 8, 9, 10, 16, 17, e 18	96,25%
8	1, 6, 8, 10, 12, 15, 16, e 17	96,25%
8	1, 4, 10, 12, 13, 14, 15, e 16	96,25%
8	1, 6, 8, 10, 13, 15, 17, e 18	96,25%
8	2, 5, 7, 8, 9, 10, 14, e 16	96,25%
8	1, 2, 7, 9, 10, 11, 12, e 14	96,25%
8	5, 6, 8, 10, 15, 16, 17, e 18	96,25%
8	8, 9, 10, 11, 12, 13, 15, e 18	96,25%
10	1, 4, 6, 7, 9, 10, 11, 12, 13, e 18	96,25%
10	1, 2, 4, 7, 8, 10, 11, 12, 16, e 18	96,25%
10	3, 4, 7, 10, 11, 14, 15, 16, 17, e 18	96,25%
10	3, 5, 6, 9, 10, 11, 13, 14, 15, e 17	96,25%
10	1, 4, 6, 7, 10, 11, 12, 15, 17, e 18	96,25%
10	4, 5, 6, 7, 10, 11, 14, 15, 16, e 18	96,25%
10	1, 3, 5, 6, 10, 11, 12, 15, 16, e 17	96,25%
10	1, 5, 6, 8, 9, 10, 12, 15, 16, e 18	96,25%
10	1, 2, 8, 9, 10, 11, 12, 14, 15, e 16	96,25%
10	2, 4, 5, 7, 9, 10, 12, 13, 15, e 16	96,25%
10	1, 2, 5, 8, 10, 11, 13, 14, 15, e 18	96,25%
10	1, 4, 5, 7, 8, 9, 10, 12, 16, e 18	96,25%
10	1, 2, 3, 4, 7, 9, 10, 12, 16, e 17	96,25%
10	1, 2, 5, 6, 8, 9, 10, 11, 15, e 18	96,25%

Na Tabela 13 são apresentadas algumas das variáveis que, no mínimo, mantêm a precisão da rede neural de duas camadas cuja primeira camada e camada de saída são compostas por neurônios que usam a função de ativação *Softmax* e, na segunda camada, neurônios sigmoidais. Pode-se observar que no teste de variáveis de entrada desta rede, a única topologia otimizada é a de quatro variáveis de entrada, enquanto as outras se mantêm inalteradas se comparadas a Tabela 6.

Tabela 13 – variáveis otimizadoras da Rede Neural (*Softmax* – Sigmoide – *Softmax*).

.

Nº de entradas	***variáveis***	***Precisão***
4	1, 10, 14, e 18	96,25%
4	10, 13, 15, e 16	96,25%
4	5, 10, 13, e 18	96,25%
4	10, 13, 17, e 18	96,25%
4	8, 10, 12, e 18	96,25%
6	10, 11, 13, 15, 16, e 18	96,25%
6	4, 9, 10, 11, 15, e 16	96,25%
6	1, 8, 10, 14, 15, e 18	96,25%
6	2, 5, 10, 14, 15, e 18	96,25%
8	2, 3, 4, 7, 10, 13, 14, e 17	96,25%
8	2, 5, 9, 10, 12, 13, 16, e 17	96,25%
8	1, 2, 6, 9, 10, 11, 12, e 17	96,25%
8	1, 2, 5, 6, 10, 11, 15, e 16	96,25%
8	5, 8, 9, 10, 11, 12, 17, e 18	96,25%
8	1, 6, 8, 10, 11, 12, 13, e 18	96,25%
8	1, 10, 11, 13, 14, 15, 17, e 18	96,25%
8	2, 4, 6, 7, 10, 12, 14, e 17	96,25%
8	8, 10, 12, 13, 14, 15, 16, e 17	96,25%
8	6, 8, 10, 11, 12, 13, 15, e 16	96,25%
10	2, 3, 4, 7, 10, 13, 14, 16, 17 e 18	96,25%

Na Tabela 14 são apresentadas algumas das variáveis que, no mínimo, mantêm a precisão da rede neural de uma camada cuja primeira camada é composta por neurônios sigmoidais e na camada de saída por neurônios que usam a função de ativação *Softmax*. Ao observar o teste de combinação de variáveis para esta rede, podemos avaliar que estão inalteradas se comparados com o da Tabela 8.

Tabela 14 – variáveis otimizadoras da Rede Neural (Sigmoide – *Softmax*).

Nº de entradas	***variáveis***	***Precisão***
4	1, 7, 10, e 14	96,25%
4	7, 10, 11, e 15	96,25%
4	3, 8, 10, e 12	96,25%
6	2, 5, 7, 8, 10, e 12	96,25%
6	2, 8, 10, 12, 13, e 16	96,25%
6	6, 10, 11, 13, 16, e 18	96,25%
6	1, 2, 5, 10, 11, e 15	96,25%
6	5, 8, 9, 10, 14, e 17	96,25%
6	1, 6, 9, 10, 14, e 15	96,25%
6	3, 8, 10, 12, 14, e 16	96,25%
6	4, 9, 10, 15, 16, e 17	96,25%
6	1, 9, 10, 12, 13, e 18	96,25%
6	3, 9, 10, 14, 16, e 17	96,25%
8	2, 7, 8, 9, 10, 11, 12, e 13	96,25%
10	2, 3, 7, 10, 11, 12, 14, 15, 17, e 18	96,25%
10	1, 3, 6, 8, 10, 12, 13, 16, 17, e 18	96,25%
10	1, 2, 7, 8, 10, 12, 15, 16, 17, e 18	96,25%
10	8, 9, 10, 11, 12, 13, 14, 15, 17, e 18	96,25%
10	6, 8, 9, 10, 11, 13, 15, 16, 17, e 18	96,25%

Na Tabela 15 são apresentadas algumas das variáveis que, no mínimo, mantêm a precisão da rede neural de uma camada cuja primeira camada e camada de saída são compostas de neurônios que usam a função de ativação *Softmax*. No teste de combinação desta rede, todas as topologias (que concernem o número de variáveis de entrada) foram otimizadas se comparadas com as dos testes de encontro de topologias com variáveis de estado fixa da Tabela 7, mostrando assim a fundamental importância de se achar a combinação correta de variáveis de entrada que melhor classificam o problema.

Tabela 15 – variáveis otimizadoras da Rede Neural (*Softmax* – *Softmax*).

Nº de entradas	***variáveis***	***Precisão***
4	6, 10, 12, e 14	96,25%
4	8, 10, 13, e 16	96,25%
4	1, 8, 10, e 13	96,25%
4	2, 10, 14, e 16	96,25%
4	6, 10, 14, e 15	96,25%
4	10, 12, 14, e 17	96,25%
4	8, 10, 12, e 15	96,25%
6	2, 8, 10, 13, 16, e 18	96,25%
6	2, 3, 10, 12, 15, e 17	96,25%
6	2, 4, 7, 10, 12, e 15	96,25%
8	5, 6, 7, 8, 9, 10, 13, e 14	96,25%
8	2, 5, 10, 12, 13, 16, 17, e 18	96,25%
8	1, 5, 9, 10, 11, 12, 15, e 16	96,25%
10	2, 3, 4, 5, 9, 10, 11, 12, 15, e 17	95,00%
10	1, 2, 3, 4, 5, 7, 9, 10, 11, e 17	95,00%

Na Tabela 16 são apresentadas algumas das variáveis que otimizam a rede neural sem camadas ocultas cuja camada de saída é composta por neurônios que usam a função de ativação *Softmax*. Como a rede não tem camadas ocultas, só existe uma topologia possível para cada número de dados entrada, portanto, a rede foi otimizada para todas a topologias estudadas no Teste 1, como pode ser observado se os resultados forem comparados aos da Tabela 9.

Tabela 16 – variáveis otimizadoras da Rede Neural (*Softmax*).

Nº de entradas	***variáveis***	***Precisão***
4	10, 12, 13, e 16	95,00%
6	7, 8, 10, 12, 13, e 15	96,25%
6	10, 11, 12, 14, 15, e 16	96,25%
8	4, 7, 10, 11, 12, 13, 14, e 15	96,25%
10	1, 3, 4, 8, 10, 12, 14, 15, 16, e 18	96,25%

Neste estudo também foi feita uma análise das variáveis que mais influenciam a precisão da rede neural. Essa análise foi realizada utilizando todas as combinações de variáveis de entrada possíveis para conjuntos de 4 variáveis e somando a precisão da rede a cada variável testada, podendo assim observar quais variáveis estiveram em

treinamentos que obtiveram precisões maiores. A Figura 16, apresenta-se o gráfico de barras do teste.

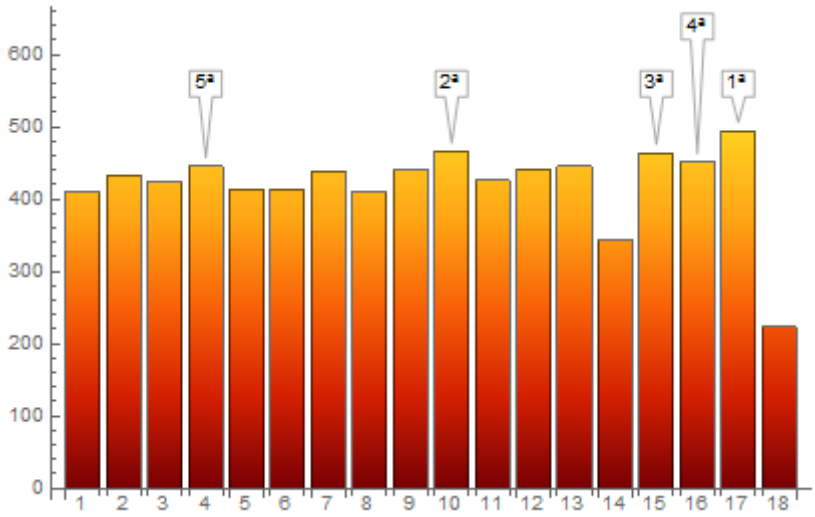

Figura 16 – Gráfico de barras da relevância das variáveis de estado.

O gráfico de barras da Figura 16 foi uma maneira quantitativa de se observar as variáveis que estavam envolvidas nos testes com maiores precisões. Apesar das colunas serem próximas, o padrão do gráfico se repete, mostrando que as variáveis de estado que mais aparecem em testes com alta precisão são:

1ª. Variável 17 - Nível do pressurizador (%);
2ª. Variável 10 - Vazão de ruptura (kg/s);
3ª. Variável 15 - Potência nuclear (%);
4ª. Variável 16 - Margem de sub-resfriamento (°C);
5ª. Variável 4 - Vazão no núcleo (kg/s);

CAPÍTULO 6
CONCLUSÕES

O presente trabalho tem como objetivo a criação de um sistema de identificação de acidentes de uma instalação nuclear via redes neurais que identificasse com precisão suficientemente alta os acidentes SGTR, LOCA e BLACKOUT, além de identificar o cenário de operação normal de uma central nuclear. Além disso, o trabalho buscou encontrar a melhor topologia de rede para maximizar a precisão das classificações dos cenários de uma central nuclear, visando minimizar a demanda cognitiva dos operadores na sala de controle.

Para atingir esse objetivo, foram implementadas redes neurais na linguagem de programação Python utilizando a biblioteca de código aberto TensorFlow. Essas redes neurais tiveram seus parâmetros testados de forma automatizada para a aquisição dos resultados e análise de dados.

Para verificação do método desenvolvido, foram feitos dois testes: o primeiro, teve como objetivo obter a melhor topologia para redes neurais com 4, 6, 8 e 10 neurônios de camada de entrada, com número de camadas ocultas igual a 0, 1 ou 2 utilizando as funções de ativação; sigmoide ou *Softmax.* O segundo teste teve como objetivo usar as melhores topologias, funções de ativação e número de entradas da rede selecionadas no teste 1 para encontrar o menor número de variáveis de estado, que se pode usar como entrada da rede para obter a melhor precisão.

No primeiro teste, foram obtidas as topologias com maior precisão e menor número de conexões sinápticas. Todas as topologias selecionadas tiveram bons resultados, se mantendo em sua grande maioria com precisão a cima de 90,00%, com destaque especial as topologias que em sua totalidade foram treinadas por funções *Softmax* que obtiveram resultados mais elevados. Incluindo o melhor deles, para a topologia de duas camadas ocultas, com funções *Softmax* nos neurônios de todas as camadas da rede, na qual, na camada de entrada o número de neurônio foi 4, e na primeira e segunda camada foram respectivamente 7 e 5, resultando em uma precisão de 97,50%.

No segundo teste, as topologias previamente selecionadas no primeiro teste foram testadas usando conjuntos de variáveis de estado distintos. O melhor resultado da rede foi obtido com o conjunto de 4 variáveis (4, 7, 10 e 17) que são respectivamente a Vazão do Núcleo(Kg/s), Pressão do Gerador de Vapor(Mpa), Vazão de Ruptura(Kg/s) e Nível do Pressurizador(%) e obteve uma precisão de 97,50%. Apesar de outras topologias testadas (conjuntos de variáveis) apresentarem resultados satisfatórios nenhuma obteve a precisão de 97,50%.

E desta forma, este trabalho mostrou a viabilidade do uso de redes neurais/Python/TensorFlow e a capacidade do método de fazer classificações de cenários de acidente e operação normal de uma central nuclear com precisões superiores a 95%. Além disso, o presente trabalho conseguiu demonstrar que um conjunto de 4 variáveis é possível identificar o evento em curso.

Pode se concluir, que a implementação de um sistema de identificação de acidentes via redes neurais em Python usando a biblioteca TensorFlow é uma proeminente ferramenta para lidar com o problema de identificação de acidentes de uma central nuclear, fornecendo uma camada a mais de proteção de segurança e, por consequência diminuição da carga cognitiva do operador da sala de controle.

6.1. Trabalhos Futuros

Para trabalhos futuros, é proposto a utilização de mais tipos de acidentes na metodologia proposta.

Além disso, propõe-se estudar a contribuição de cada variável de estado para a precisão da rede neural, de forma a encontrar os parâmetros de maior relevância para a entrada da rede neural e por consequência para o monitoramento da usina no caso do tipo do acidente em curso.

REFERÊNCIAS BIBLIOGRÁFICAS

ABADI, M. et al. TensorFlow: A system for large-scale machine learning. **The advanced computing systems associations**, 2016.

ABADI, M. et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. **Distributed, Parallel, and Cluster Computing**, Março 2016.

ACTIVATION Function. **DeepAI**. Disponivel em: <https://deepai.org/machine-learning-glossary-and-terms/activation-function>. Acesso em: 26 Dezembro 2018.

ALVARENGA, M. A. B. **Diagnóstico do Desligamaneto de um Reator Nuclear através de Técnicas Avançadas de Inteligência Artificial**. [S.l.]: [s.n.], 1997.

AMINI, A. Projects. **MIT**. Disponivel em: <https://www.mit.edu/~amini/projects.html>. Acesso em: 26 Dezembro 2018.

BARTAL, Y.; LIN, J.; UHRIG, R. E. Nuclear Power Plant Transient Diagnostics Using Artificial Neural Networks that Allow Don't Know Classifications. **Nuclear Technology**, v. 110, p. 346-449, Junho 1995.

BARTLET, E. B.; UHRIG, R. E. Nuclear Power Plant Status Diagnotics Using Artificial Neural Networks. **AI91**, p. 643-653, 1991.

BASU, A.; BARTLETT, E. B. BACKIPROPAGATION ARCHITECTURE OPTIMIZATION AND AN APPLICATION IN NUCLEAR POWER PLANT DIAGNOSTICS. **Proceedings of the American Power Conference**, Abril 1993.

BENDERSKY, E. The Softmax function and its derivative. **Eli Bendersky's website**, 2016. Disponivel em: <https://eli.thegreenplace.net/2016/the-softmax-function-and-its-derivative/>. Acesso em: 18 Dezembro 2018.

BRIDLE, J. S. Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition. **NATO ASI Series (Series F: Computer and Systems Sciences)**, 1990.

CLEVERT, D.-A.; UNTERTHINER, T.; HOCHREITER, S. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs), 2016.

COMMITTEE ON LESSONS LEARNED FROM THE FUKUSHIMA NUCLEAR ACCIDENT FOR IMPROVING SAFETY AND SECURITY OF U.S. NUCLEAR PLANTS. **Lessons Learned from the Fukushima Nuclear Accident for Improving safety of U.S. Nuclear Plant**. [S.l.]: The Nacional Academy Press, 2014.

FARHADI, F. **LEARNING ACTIVATION FUNCTIONS IN DEEP NEURAL NETWORKS**. [S.l.]: [s.n.], 2017.

FAUSETT, L. **Fundamentals of Neural Network:** Architectures, Algorithms and applications. [S.l.]: [s.n.], 1994.

FURUKAWA, H.; UEDA, T.; KITAMURA, M. **Use of Self Organazing Neural Networks for Rational Definition of Plant Diagnostic Symptoms**. Proceedings of the International Topical Meeting on Computer- Based Human Support System. [S.l.]: [s.n.]. 1995. p. 441-448.

GÉRON, A. **Hands-On Machine Learning with Scikit-Learn & TensorFlow**. [S.l.]: O'REILLY, 2017.

HAHNLOSER, R. H. R. et al. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. **Nature**, 22 Junho 2000. 947-951.

HAN, J. The influence of the Sigmoid Function Parameters on the Speed of Backpropagation Learning. In: MIRA, J. **From Natural do Artificial Neural Computation**. [S.l.]: [s.n.], 1995.

HAYKIN, S. **Redes Neurais:** Pricípios e práticas. [S.l.]: Bookman, 2001.

HSIAO, T.; LIN, C.; YUANN, Y. R. Identification of initiating events for pressurized water reactor accidents. **Annals of Nuclear Energy**, 2010. 1502-1512.

HWANG, M. I.; LIN, J. W. Information dimension,information overload and decision quality. **Journal of Information Science**, p. 213-218, Outubro 1999.

IAEA. **SAFETY ASPECTS OF STATION BLACKOUT AT NUCLEAR POWER PLANTS**. [S.l.]: [s.n.], 1985.

JEONG, E.; FURUTA, K.; KONDO, S. Identification of Transient in Nunclear Power Plant Using Adaptative Template Matching with Neural Network, Proceedings of the International Topical Meeting on Nuclear Plant Intrumentation. **Control and Human Machine Interface Tecnologies**, p. 243-250, 1996.

KENNETH D. KOK. **Nuclear Engineering Handbook**. [S.l.]: CRC PRESS, 2016.

KESKAR, N. S.; SOCHER, R. Improving Generalization Performance by Switching from Adam to SGD, 2017.

KHATTAK, M. A. et al. Deterministic Safety Analysis for High Level Waste (Spent Fuel): Management of NPP, A Review. **Progress in Energy and Environment**, p. 62-74, Janeiro 2018.

KINGMA, D. P.; BA, J. **Adam:** A Method for Stochastic Optimization. 3rd International Conference for Learning Representations. [S.l.]: [s.n.]. 2015.

KOK, K. D. **Nuclear Engineering Handbook**. [S.l.]: CRC PRESS, 2016.

KWON, K. C.; KIM, J. H. Accident identification in nuclear power plants using hidden Markov models. **Engineering Applications of Artifcial Intelligence**, n. 12, p. 491-501.

LEE, Y. W. P. A. J. H. **Safety Review on Recent Steam Generator Tube Failure in Korea and Lessons**. [S.l.]: Transactions of the 17th International Conference on Structural Mechanics in Reactor Technology (SMiRT 17), 2003.

LITHUANIAN ENERGY INSTITUTE. **HANDBOOK ABOUT THE IGNALINA**. [S.l.]: [s.n.], 1997.

MACDONALD, P. E. et al. **Steam Generator Tube Failures**. [S.l.]: NUREG/CR-6365, 1996.

MARTINS, C. A. C. **Estimativa da profundida de carbonatação do concreto com o uso de redes neurais**. Recife: [s.n.], 2011.

MAZZONI, O. S. **Electrical Systems for Nuclear Power Plants**. [S.l.]: IEEE Press, 2018.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, v. 5, p. 115-133, 1943.

MEDEIROS, J. A. C. C. Enxames de Partículas como Ferramenta de Otimização em Problemas Complexos da Engenharia Nuclear. **Tese de D.Sc**, Rio de Janeiro, 2005.

MOL, A. C. A. Um Sistema de Identificação de Transientes com Inclusão de Ruídos e Identificação de Eventos Desconhecidos. [S.l.]: [s.n.], 2002.

MOL, A. C. A.; MARTINEZ, A. S.; SCHIRRU, R. A New Approach for Transient Identification with "Don't Know" Response Using Neural Networks. In: RUAN, I. D.; FANTONI, P. F. **Power Plant Surveillance and Diagnostics:** Applied Research with Artificial Intelligence. [S.l.]: [s.n.], 2002. p. 253-272.

NICOLAU, A. D. S. **Algorítmo Evolucionário de Inspiração Quântica Aplicado na Otimização de Problemas da Engenharia Nuclear**. Rio de Janeiro: [s.n.], 2014.

PALM, W. J. **System Dynamics**. 2. ed. [S.l.]: [s.n.], 2010.

PETRANGELI, G. **Nuclear Safety**. [S.l.]: Elsevier, 2006.

PINHEIRO, V. H. C.; SCHIRRU, R. Genetic Programming Applied to the Identification of Accidents of a PWR Nuclear Power Plant. **Annals of Nuclear Energy**, v. 124, p. 335-341.

POULIAKIS, A. et al. Artificial Neural Networks as Decision Support Tools in Cytopathology: Past Present and Future. **Biomedical Engineering and Computational Biology 2016**, 19 Janeiro 2016. 1-17.

RAGHEB, M. **Notas de aula do professor M. Ragheb**. [S.l.]: [s.n.], 2016.

RAUBER, P. E. Notes on Neural Networks, 2015. Disponivel em: <http://paulorauber.com/work.html>. Acesso em: 2018.

ROTH, E. M.; MUMAW, R. J. **An Empirical Investigation of Operator Performance in Cognitively Demanding Simulated Emergencies**. [S.l.]: [s.n.], 1994.

SEHGAL, B. R. **Nuclear Safety in Light Water Reactors:** Severe Accident Phenomenology. [S.l.]: Elsevier, 2012.

SHARMA, S. Activation Functions: Neural Networks. **Towards Data Science**, 2017. Disponivel em: <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>. Acesso em: 26 Dezembro 2018.

SINGER, Y. Class notes: Advanced Optimization , 2016.

SOFTMAX Regression. **Deep Learning:** Computer Science Department, Stanford University. Disponivel em: <http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/>. Acesso em: 26 Dezembro 2018.

TICHONOV, D. **https:** //medium.com/@dmitrijtichonov, 2017. Disponivel em: <https://medium.com/@dmitrijtichonov/debunking-loss-functions-in-deep-learning-4b9abc4c8d4c>. Acesso em: 11 dez. 2018.

TURING, A. M. Computing Machinery and Intelligence. **Mind**, v. 59, n. 236, p. 433-460, 1950.

VILLAS BÔAS, M. J. Diagnóstico de classe utilizando inteligência de enxames aplicado ao problema de identificação de transientes nucleares. **Tese de M.Sc**, 2011.

WEISSTEIN, E. W. Hyperbolic Tangent. **MathWorld--A Wolfram Web Resource**. Disponivel em: <http://mathworld.wolfram.com/HyperbolicTangent.html>. Acesso em: 26 Dezembro 2018.